Correcció de tweets usant word embeddings i FreeLing

Autor:
Sergi Llamas Xart

Director:
Lluís Padró Cirera
Departament de Ciències de la Computació

Titulació:
Grau en Enginyeria Informàtica
Especialitat de Computació

Universitat:
UPC – BarcelonaTech
Facultat d'Informàtica de Barcelona (FIB)

Data de defensa:
Gener de 2017

Resum

La correcció automàtica és un dels problemes oberts actualment en el camp del processament de llenguatge natural (PLN). La seva principal aplicació és la de normalitzar el text que es passarà com a entrada a processos posteriors com l'anàlisi o l'extracció de característiques. L'objectiu d'aquest projecte és desenvolupar un mòdul de correcció automàtica en castellà usant word embeddings, una tècnica utilitzada en PLN que permet obtenir representacions vectorials de les paraules d'un corpus de text. Els resultats obtinguts queden per sota dels de models tradicionals com els 3-grames, però alhora plantegen noves línies de recerca per al futur.

Resumen

La corrección automática es uno de los problemas abiertos actualmente en el campo del procesamiento de lenguaje natural (PLN). Su principal aplicación es la de normalizar el texto que se pasará como entrada a procesos posteriores como el análisis o la extracción de características. El objetivo de este proyecto es desarrollar un módulo de corrección automática en castellano usando word embeddings, una técnica utilizada en PLN que permite obtener representaciones vectoriales de las palabras de un corpus de texto. Los resultados obtenidos quedan por debajo de los de modelos tradicionales como los 3-gramas, pero al mismo tiempo plantean nuevas líneas de investigación para el futuro.

Abstract

Automatic correction is one of the open problems in the natural language processing (NLP) field. Its main application is the normalization of input text sent to subsequent processes as text feature extraction or analysis. The goal of this project is the development of a correction module in Spanish that makes use of word embeddings, a technique in NLP that can map the words from a corpus to numerical vector representations. The results obtained are below those achieved with more traditional models like 3-grams, but also suggest new ideas for future research.

Agraïments

Vull agrair a en Lluís Padró per tota l'ajuda, les discussions tècniques i la bona disposició mostrada durant tot el projecte. Sense la seva contribució tant com a director del treball com a desenvolupador de FreeLing, aquest treball no hauria estat possible.

També vull agrair a tots els meus companys d'universitat que d'una manera o altra m'han ajudat a arribar fins a aquí, i que sempre han estat disposats a discutir idees i donar un cop de mà quan ho he necessitat.

Índex

Índex **4**

1 Abast i contextualització **7**
1.1 Breu introducció al problema 7
1.2 Objectiu del projecte 8
1.3 Contextualització 8
1.4 Abast del projecte 11

2 Planificació temporal **15**
2.1 Descripció de les tasques 15
2.2 Taula temporal 18
2.3 Diagrama de Gantt 18
2.4 Valoració d'alternatives 19
2.5 Recursos usats 19
2.6 Pla d'acció 20

3 Pressupost i sostenibilitat **23**
3.1 Gestió econòmica 23
3.2 Lleis i regulacions 26
3.3 Sostenibilitat 27

4 Revisió de la planificació **31**
4.1 Canvis en la metodologia 31
4.2 Canvis en la planificació 31

5 Estructura d'un corrector automàtic **35**
5.1 Pipeline bàsic 35
5.2 Descripció de les parts 35
5.3 Complexitat del problema 43

6 Word embeddings **45**
6.1 Models usats en l'entrenament 45
6.2 Representació vectorial de les paraules 47
6.3 Entrenament 50
6.4 Propietats 51
6.5 Word2Vec 53

7 Implementació del projecte **55**
7.1 Word embeddings 55
7.2 Extracció de dades per proves i entrenaments 59
7.3 Mòdul de correcció 61
7.4 Altres canvis a FreeLing 70
7.5 Programes de prova 70

8 Anàlisi dels resultats **73**
8.1 Primer joc de proves 73
8.2 Comparació amb Tweet-Norm 74

9 Conclusions **79**

Bibliografia **81**

1. *Abast i contextualització*

1.1 Breu introducció al problema

El processament de llenguatge natural (també abreviat com a PLN) és el camp d'estudi que tracta d'analitzar i extreure informació del llenguatge natural a partir de tècniques computacionals. Degut a la complexitat d'aquesta tasca, molts dels problemes tractats en aquest camp encara no tenen solucions òptimes o comparables en mitjana a les que podrien oferir els propis humans: la correcció automàtica de textos n'és un dels exemples i el cas que ens ocupa.

D'una banda, les diverses tecnologies que tracten de resoldre aquest problema ja són habituals a la nostra societat avui dia. Exemples en són els sistemes de correcció automàtica integrats en processadors de textos o telèfons intel·ligents, que milions de persones usen diàriament.

Tot i això, hi ha altres usos per a la correcció automàtica. L'extracció d'informació de textos n'és un bon exemple. Avui en dia podem trobar grans quantitats de text a internet que poden resultar d'interès per a tot tipus de propòsits. En aquests casos, on les fonts poden ser de qualitat variable i presentar una quantitat important d'errors ortogràfics i d'altres tipus, la correcció automàtica esdevé un pas fonamental per preparar les dades abans de les anàlisis finals.

És en aquest context que pensem que noves tècniques d'aprenentatge automàtic com els word embeddings podrien ser aplicades per a tractar de millorar, en diferents aspectes, les solucions proposades fins al moment per a certes parts del procés de la correcció automàtica de textos.

1.2 Objectiu del projecte

L'objectiu del projecte és desenvolupar un mòdul capaç de realitzar la correcció automàtica i normalització de tweets[1] en castellà usant word embeddings. Aquest projecte serà implementat sobre FreeLing[2] [17] i n'aprofitarà d'altres mòduls ja implementats per a realitzar moltes de les tasques de processament de llenguatge natural requerides.

Els word embeddings són un model de llenguatge que permet representar les paraules d'un idioma determinat com a vectors d'un mateix espai multidimensional. Aquesta tècnica s'utilitzarà per a optimitzar la selecció de les millors correccions per a cada paraula no reconeguda trobada en un tweet, tenint en compte també la resta de paraules de la frase.

1.3 Contextualització

Termes i conceptes

Aprofitem aquesta secció per veure les definicions més completes i formals d'alguns dels conceptes que fins ara s'han mencionat únicament de passada:

- **Correcció automàtica de textos**: conjunt de tècniques utilitzades per a la correcció automatitzada dels errors existents en un text. Aquest projecte es centrarà exclusivament en els errors ortogràfics, tot i que en el sentit més ampli de l'expressió la correcció automàtica de textos pot incloure també la correcció d'altres tipus d'errors, com poden ser els morfosintàctics. Aquesta correcció automàtica també és anomenada normalització.
- **Model de llenguatge**: distribució de probabilitat sobre una seqüència de paraules. Donada una seqüència de mida m, la probabilitat de la seqüència completa és $P(w_1, \ldots, w_m)$.
- **Bosses de paraules**: de *bag-of-words* en anglès. És una representació d'un conjunt de paraules sense tenir-ne en compte l'ordre.
- **N-grames**: model de llenguatge que calcula la probabilitat d'una seqüència de N paraules, amb les paraules en un ordre específic. Els n-grames són importants en el projecte ja que són un dels mètodes tradicionalment usats per determinar quina paraula d'una llista de candidates encaixa millor en un determinat context.

[1] Els tweets són els missatges breus publicats pels usuaris de la plataforma social Twitter.

[2] FreeLing és una eina de processament de llenguatge natural multilingüe i de codi obert, accessible a `http://nlp.lsi.upc.edu/freeling/node/1`.

- **word embeddings**: conjunt de tècniques i models del llenguatge que es basen en la representació de les paraules com a vectors de dimensionalitat reduïda respecte la mida del vocabulari. Aquesta representació, a més, és obtinguda amb mètodes d'aprenentatge automàtic basats en xarxes neuronals. Els word embeddings van ser proposats inicialment el 2003 per Yoshua Bengio [2], tot i que la seva popularitat realment no es va disparar fins que al 2013 Tomas Mikolov i el seu grup d'investigació van publicar un nou article mostrant significatives millores en els resultats obtinguts [12] i oferint una nova implementació en codi obert sota el nom de Word2Vec[3].

Usuaris del producte

Els principals beneficiats del treball realitzat al projecte, a priori, seran els usuaris de FreeLing, i per tant, indirectament també els usuaris de qualsevol aplicació o servei que utilitzi FreeLing i els mòduls relacionats amb els word embeddings.

Cal aclarir que la integració dels word embeddings a FreeLing ja suposa, per sí mateixa, un benefici important per a qualsevol aplicació o servei que utilitzi PLN. Els word embeddings són una tècnica que, com ja s'ha explicat, permet obtenir representacions vectorials de les paraules. Això vol dir que més enllà de la correcció automàtica, aquestes representacions poden tenir moltes altres possibles aplicacions dins el camp del PLN. L'elaboració de resums extractius [10], el càlcul de similitud entre paraules i el completat de frases [14] en són exemples contrastables. Tenir aquest model integrat a FreeLing, per tant, en facilitaria el seu ús a altres investigadors i desenvolupadors.

Si ens centrem de nou en la correcció automàtica, els beneficis i beneficiats canvien lleugerament. D'una banda, el projecte és interessant per als investigadors des del punt de vista de la recerca, ja que oferirà resultats sobre l'efectivitat de l'ús dels word embeddings en la normalització de textos en castellà.

Tanmateix, aquest mètode pot presentar diferències importants respecte d'altres mètodes com els n-grames que poc tenen a veure amb la qualitat dels resultats obtinguts. És el cas de la memòria requerida per a l'execució dels diferents mètodes. Els menors requeriments de memòria per part dels word embeddings respecte els 3-grames, per exemple, suposen que fins i tot amb resultats de qualitat lleugerament inferior, els word embeddings podrien resultar particularment interessants per al desenvolupament de sistemes de correcció automàtica en plataformes amb recursos limitats.

[3] `https://code.google.com/archive/p/word2vec/`

Estat de l'art

Tal com ja s'ha detallat a la secció de termes i conceptes, els word embeddings són una tècnica relativament recent que s'ha utilitzat amb resultats satisfactoris en diverses tasques relacionades amb el PLN.

Ja amb la primera implementació de Word2Vec trobem un precedent d'ús molt similar: el *Microsoft Research Sentence Completion Challenge* [5]. Tomas Mikolov demostra en un article del 2013 [14] com l'ús dels word embeddings permet millorar els que fins aleshores eren els resultats de l'estat de l'art, passant d'un 55.4% a un 58.9% de precisió en la tasca de completar les frases del conjunt de dades presentat per Microsoft.

Aquest precedent és molt similar ja que aborda la tasca de seleccionar la paraula que millor encaixa en un context determinat. La diferència principal del treball de Mikolov respecte a aquest projecte és que la llista de paraules candidates no ve donada externament com passa en el cas del conjunt de dades de Microsoft, sinó que es calcula exclusivament usant mètodes computacionals sense cap filtrat ni intervenció humana posterior. A més, no s'ha de seleccionar una única paraula per frase, sinó que en una frase poden haver-hi un nombre arbitrari de paraules mal escrites que requereixin correccions.

Respecte a la tasca completa de correcció automàtica de textos, ens trobem dos escenaris diferents. En primer lloc trobem l'existència de les grans empreses que disposen de productes comercials per a diferents plataformes. En aquests casos no acostumem a tenir mètriques, implementacions en codi obert, articles ni dades oficials que ens permetin evaluar ni comparar fàcilment resultats. A més, els algoritmes utilitzats són en la majoria de casos desconeguts. Addicionalment, també cal tenir en compte que els resultats obtinguts per a diferents llengües poden ser potencialment molt diferents. Una part fonamental de la correcció automàtica requereix tècniques estretament lligades a les propietats de la llengua per a la qual s'està desenvolupant el corrector, i fins i tot en els casos en que s'usa aprenentatge automàtic, la qualitat de les bases de dades i corpus disponibles pot arribar a ser un factor que faci variar la qualitat del resultat final.

Tot i això també hi ha un segon escenari com és el de les competicions, tallers i implementacions en codi obert que tracten de resoldre el problema. És en aquest segon escenari que trobem Tweet-Norm [1], un taller realitzat al 2013 en que diferents participants oferien les seves solucions per al mateix escenari que ocupa aquest projecte: la correcció i normalització de tweets en castellà. En aquest taller els millors resultats, obtinguts per Jordi Porta i José Luis Sancho [18] utilitzaven 3-grames a l'hora de resoldre el problema de la seqüència més probable de paraules.

En anglès en canvi trobem el *Workshop on Noisy User-generated Text*[4] (abreviat W-NUT), on els millors resultats obtinguts utilitzaven una combinació de n-grames i

[4] `http://noisy-text.github.io/2015/`

salt-grames (*skip-grams* en anglès) [9], entre d'altres, on els salt-grames consisteixen en un model similar als n-grames, però que en lloc d'utilitzar grups de paraules consecutives es "salten"o ometen algunes de les paraules.

1.4 Abast del projecte

L'abast del projecte inclou els següents punts:

- Integració de funcionalitats de Word2Vec —una implementació en codi obert dels word embeddings— a la llibreria de FreeLing.
- Entrenament del model de word embeddings amb textos correctes extrets de diverses fonts.
- Implementació d'un mòdul capaç de seleccionar la millor combinació de paraules a partir del model de llenguatge entrenat i d'altres algoritmes.
- Anàlisi dels resultats obtinguts i comparació amb d'altres models de llenguatge.

Cal fer notar que el procés de normalització de text es divideix fonamentalment en tres etapes: preprocessat del text, generació de llistes de correccions possibles per a les paraules incorrectes i finalment selecció de les correccions més adequades.

El projecte es concentra en l'última d'aquestes etapes, i per tant les dues primeres no s'implementaran des de zero, sinó que es resoldran usant mòduls ja existents a FreeLing. La seva optimització queda per tant fora l'abast d'aquest projecte, tot i que efectivament la qualitat dels resultats obtinguts en aquestes etapes té una influència significativa en els resultats finals de la normalització.

Metodologia i rigor

Mètode de treball i possibles obstacles

Donat el termini de temps relativament curt en el que s'haurà de desenvolupar el projecte, un dels majors problemes que es pot donar és el de desenvolupar grans parts de codi que al final resulten tenir errors o s'han de descartar per culpa de problemes de disseny o situacions inesperades. Així doncs, considero que una part important consisteix en desenvolupar i provar el codi en la mesura del possible de forma freqüent. Aquesta metodologia es correspon a l'anomenat desenvolupament incremental i iteratiu.

Per la resta, considero que els únics problemes importants que poden haver-hi són de terminis. Serà necessari mantenir un treball constant i d'acord amb el calendari previst per tal de poder assegurar que el resultat final està ben acabat i no calen modificacions o canvis d'última hora.

Eines de seguiment

Tal com ja s'ha esmentat prèviament, mantenir un calendari estricte i una planificació realista és molt important en aquest tipus de projectes. A més, es mantindran reunions regularment amb el director del projecte en funció de la necessitat i els requeriments de cada etapa del treball.

Mètode de validació

El mètode de validació per al projecte és el mateix que s'ha detallat en els seus objectius. Cal comparar els resultats d'usar els word embeddings davant els resultats obtinguts amb altres models de llenguatge. Els n-grames, i més concretament els 3-grames, són un dels mètodes tradicionals que més bons resultats ofereixen, com ja s'ha vist a la secció de l'estat de l'art.

Per tal d'aconseguir això es poden comparar els resultats obtinguts amb una implementació pròpia dels n-grames i/o amb els resultats obtinguts pels diferents grups que van participar al Tweet-Norm l'any 2013.

Límits del projecte

L'objectiu bàsic del projecte és el d'avaluar els resultats d'usar els word embeddings davant altres models de llenguatge en l'àmbit de la correcció automàtica i normalització de text en castellà.

Tenint en compte que FreeLing és una eina que permet realitzar PLN sobre múltiples llengües diferents, la integració del codi de Word2Vec també haurà de permetre aquest multilingüisme en la mesura del possible. Vist això, és fàcil pensar que seria interessant no avaluar només el mètode en una llengua, sinó tractar d'obtenir resultats reals en múltiples llengües, com podrien ser el castellà i el català. Tècnicament això és assequible, i podria ser una extensió raonable del projecte, però a priori aquest no serà un objectiu del treball. Això és així perquè a l'hora de la veritat no es requereixen únicament els programes, sinó també els corpus de text, bases de dades i entrenaments dels models utilitzats. Òbviament disposar dels programes suposa una ajuda molt important, però la decisió de provar una o més llengües s'haurà de prendre quan el projecte es trobi en una etapa més avançada i es conegui millor l'esforç i treball addicional requerit per a provar aquestes alternatives.

Un altre dels punts en els que a priori el treball no entrarà en profunditat és en les diferències d'ús de memòria entre models de llenguatges diferents. Aquestes diferències molt probablement es mesuraran i detallaran en la memòria final del projecte, però no s'aprofitaran per a desenvolupar directament cap sistema en plataformes específiques com podrien ser telèfons intel·ligents sense connexió a internet.

Finalment, cal esmentar que existeixen una gran quantitat d'optimitzacions possibles que podrien implementar-se sobre un o més dels múltiples algoritmes i tècniques que s'utilitzaran en aquest projecte, que a més podrien variar significativament en complexitat. Els objectius inicials d'aquest projecte no contemplen, d'entrada, cap optimització particular ni cap punt del projecte especialment dedicat a l'optimització genèrica. En cas que finalment es decideixi aplicar alguna optimització, aquesta decisió serà presa durant el transcurs del projecte i en funció dels resultats preliminars que s'estiguin obtenint.

Prestacions del producte final

El producte final serà un mòdul integrat a FreeLing que permetrà la normalització de tweets basant-se en altres mòduls ja existents de FreeLing i la nova integració dels word embeddings a la llibreria. D'aquesta manera, el producte final permetrà la normalització de tweets de forma molt similar al que es va veure al Tweet-Norm del 2013, amb les òbvies diferències corresponents als mètodes computacionals empleats per a resoldre el problema.

Aquest programa final integrat a FreeLing es podrà executar des de línia de comandes i requerirà tant un model de llenguatge basat en word embeddings prèviament entrenat com el fitxer de text d'entrada a normalitzar. El resultat serà el mateix text d'entrada un cop normalitzat, amb potencialment múltiples opcions per mostrar d'altres aspectes de l'execució dels algoritmes implicats com poden ser la llista de paraules suggerides per cada paraula incorrecta, les similituds entre paraules calculades amb els word embeddings, i d'altres possibles mètriques d'interès.

2. *Planificació temporal*

2.1 Descripció de les tasques

En aquest apartat es descriuen breument les tasques principals del projecte en l'ordre aproximat en el que es realitzaran.

Planificació del projecte

La planificació del projecte, també anomenada fita inicial, és l'etapa en la que ens trobem actualment. Els punts principals d'aquesta planificació són els següents:

- Definició de l'abast i contextualització
- Planificació temporal
- Gestió econòmica i sostenibilitat
- Presentació i documentació final

Durant aquesta etapa, a més, un altre punt important és la investigació i recerca de l'estat de l'art de les tècniques i sistemes relacionats amb el projecte. Aquest punt de fet ja va començar a tractar-se abans de l'inici del curs de GEP, quan es va decidir el tema del projecte, i probablement continuarà i s'estendrà a mida que es passi a tractar amb major detall cadascuna de les tècniques i algoritmes del projecte.

Configuració i familiarització amb l'entorn de treball

Com és habitual en la majoria de projectes de computació, un dels primers temes a tractar és la configuració de les eines i la resta de software que s'utilitzarà. Aquest punt, depenent de les eines utilitzades, pot requerir un temps important, que cal tenir en compte durant l'etapa de planificació.

En el cas d'aquest projecte, concretament, hi ha diverses peces de software particularment importants que cal configurar i amb les que cal familiaritzar-se. Els casos més importants són FreeLing i Word2Vec.

En primer lloc, cal especificar que durant el projecte també s'utilitzaran altres eines, sistemes i llenguatges, com pot ser el cas de l'API de Twitter. Tot i això, menciono FreeLing i Word2Vec en especial ja que són peces de software en les que el projecte aprofundirà considerablement, i que a més estan relacionades amb el camp del processament de llenguatge natural, un camp que no es tracta durant la resta del grau en enginyeria informàtica. Així doncs, si bé es cert que també es requerirà cert temps per a aprendre a utilitzar l'API de Twitter, considero que el temps requerit per adaptar-se a noves disciplines a través de diverses eines informàtiques serà molt més important en el projecte que el temps requerit per adaptar-se a les pròpies eines informàtiques. Òbviament, el temps necessari per a la configuració de les diferents eines encara segueix present.

Desenvolupament del projecte

En aquest apartat es detallen les diverses etapes més importants durant el desenvolupament del codi i el cos principal del projecte.

Integració dels word embeddings a FreeLing

Aquesta etapa consisteix bàsicament a elaborar un mòdul preparat per treballar amb word embeddings a FreeLing. Existeixen implementacions de codi obert com word2vec que es poden utilitzar com a referència o fins i tot portar-ne algunes parts directament a FreeLing.

Obtenció de dades i entrenament del model

Un cop els word embeddings hagin estat integrat a FreeLing, tal com s'ha suggerit a l'apartat anterior, cal comprovar que s'obtenen resultats correctes. Aquest pas és complex ja que requereix l'entrenament de models implementats amb dades reals. Aquestes dades, grans quantitats de textos no formals, han de ser obtingudes i seleccionades durant el propi projecte. De fet sí existeixen algunes bases de dades públiques que contenen aquest tipus d'informació, però únicament en quantitats reduïdes, que si bé es poden fer servir per a realitzar les primeres proves, probablement seran insuficients per a aconseguir un entrenament del model de qualitat. És en aquesta etapa del projecte on es farà un ús més important de l'API de Twitter.

Implementació del mòdul de normalització

Un cop finalitzada la integració del model dels word embeddings a FreeLing, encara es requereix la implementació d'altres algoritmes per tal de poder realitzar la tasca de normalització. D'aquesta manera, caldrà implementar un mòdul que faci ús tant d'eines ja existents a FreeLing com de noves funcions que permetin, entre d'altres, localitzar les paraules incorrectes en un text, obtenir les seves possibles correccions o seleccionar la combinació òptima de correccions que resulti en una reconstrucció del text el més coherent possible.

Implementació dels models a comparar

Un cop funcional la normalització de text usant word embeddings, cal preparar també els altres models de llenguatge amb els que es compararan resultats. És el cas dels n-grames, o més concretament, dels 3-grames que probablement s'acabaran utilitzant. En aquest model la quantitat de memòria necessària per a emmagatzemar tota la informació és molt significativa, ja que la quantitat de 3-grames per un idioma concret (possibles combinacions de tres paraules en seqüència) explota ràpidament. Això implica que durant la implementació d'aquest sistema probablement es requereixin tècniques d'indexació per a l'extracció eficient de dades del disc dur, entre d'altres optimitzacions, que poden requerir un esforç d'implementació considerable tot i tractar-se d'un model de llenguatge aparentment senzill.

Anàlisi dels resultats

Un cop implementades totes les parts, cal analitzar i comparar els resultats. De nou, aquesta etapa és no trivial ja que probablement requereixi el re-entrenament dels models o la obtenció de dades de prova addicionals un cop s'hagin observat els primers resultats.

Tasca final

La tasca final consistirà en l'elaboració de la documentació final, l'acabament de la memòria i la preparació de la presentació del projecte.

2.2 Taula temporal

A continuació es presenta la taula temporal amb les tasques exposades en les seccions prèvies.

Tasca	**Duració aproximada**
Planificació del projecte	80
Configuració i familiarització amb l'entorn de treball	20
Integració de word embeddings a FreeLing	100
Obtenció de dades i entrenament del model	60
Implementació del mòdul de normalització	80
Implementació dels models a comparar	90
Anàlisi dels resultats	20
Tasca final	50
Total	**500**

2.3 Diagrama de Gantt

Task name	Start date	End date
TFG	19/09/16	20/01/17
Planificació del projecte	19/09/16	17/10/16
Definició de l'abast i contextualització	19/09/16	27/09/16
Planificació temporal	27/09/16	03/10/16
Gestió econòmica i sostenibilitat	03/10/16	10/10/16
Presentació i documentació final	10/10/16	17/10/16
Configuració i familiarització amb l'entorn de treball	17/10/16	22/10/16
Integració de Word2Vec a FreeLing	22/10/16	15/11/16
Obtenció de dades i entrenament del model	16/11/16	30/11/16
Implementació del mòdul de normalització	30/11/16	17/12/16
Implementació dels models a comparar	17/12/16	05/01/17
Anàlisi dels resultats	08/01/17	13/01/17
Tasca final	13/01/17	20/01/17

Figura 2.1: Taula amb les dates aproximades d'inici i fi de les tasques.

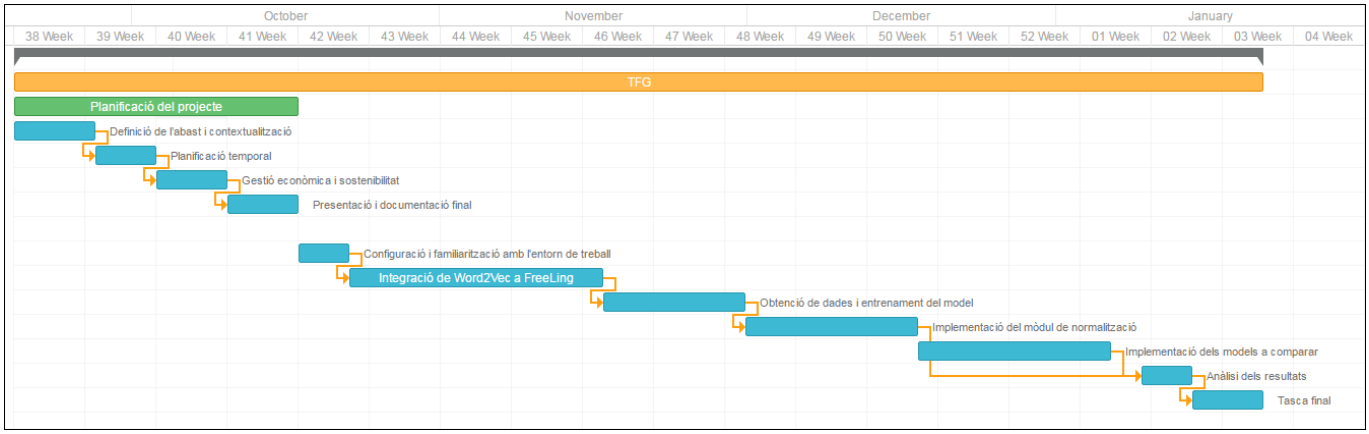

Figura 2.2: Diagrama de Gantt.

2.4 Valoració d'alternatives

Com en tot projecte de computació, avaluar la complexitat del problema a priori pot ser molt complicat, i en ocasions es poden presentar obstacles o desviacions respecte la planificació, especialment en l'etapa d'implementació del projecte.

En cas que aquestes desviacions es donin, el que es farà serà dedicar hores addicionals al projecte durant les següents setmanes a fi de recuperar el temps invertit en tractar amb les desviacions. En els pitjors dels casos, una altra mesura possible seria la de retallar temps deixant de banda la implementació multilingüe dels algoritmes utilitzats en el projecte i centrant-me exclusivament en el castellà, tot i que això seria poc desitjable.

Finalment, també seria possible aconseguir temps reduint el temps dedicat a l'extracció de textos d'entrenament, tot i que l'ideal seria tenir un gran volum de dades de qualitat i fer diverses proves per tal de poder determinar exactament quina és la quantitat de dades necessàries per a obtenir un entrenament òptim dels word embeddings en castellà.

2.5 Recursos usats

En aquesta secció s'enumeren i expliquen breument els recursos necessaris per al projecte, tan a nivell de software com de hardware.

Hardware

A nivell de hardware el projecte únicament requereix l'ús d'ordinadors personals. Més específicament, els principals ordinadors utilitzats seran els que faig servir regularment per treballar:

- Ordinador de sobretaula (AMD Athlon II X4 651 Quad-Core 3.00 GHz, 8.00 GB memòria RAM)
- Portàtil (Intel Core i3-4005U 1.70 GHz, 4.00 GB memòria RAM)

Software

- FreeLing: llibreria sobre la qual s'implementarà el projecte.
- Word2Vec: implementació en codi obert dels word embeddings.
- API de Twitter: usada en l'extracció de dades necessàries per l'entrenament dels models de llenguatge comparats en el projecte.
- Linux, Ubuntu: principal sistema operatiu sobre el qual es desenvoluparà el projecte.
- Git: sistema de control de versions de codi.

2.6 Pla d'acció

Tal com es pot observar al diagrama, la majoria de tasques són bastant seqüencials. El pla d'acció consisteix a executar-les d'aquesta mateixa manera sempre que sigui possible, però sempre sent realistes amb els terminis. Una de les primeres tasques de la implementació del projecte ja és la pròpia integració dels word embeddings a FreeLing. Això implica treballar en profunditat amb els principals sistemes del projecte des de ben aviat. En aquesta etapa ja poden aparèixer els primers obstacles i problemes inesperats per al desenvolupament del projecte, però com a mínim el propi fet de veure'ls tan aviat ja és un punt positiu. Donat el cas, tindria marge suficient com per a modificar la planificació per ajustar-la a les noves necessitats.

En general, considero aquest últim punt una part bàsica del pla d'acció: la planificació és una guia fonamental que cal seguir en el projecte, però si en algun moment deixa de ser realista o representar adequadament les necessitats del projecte, el primer que cal fer és revisar-la i modificar-la per a que torni a ser vigent.

A nivell del temps requerit pel projecte, un dels problemes es que totes les tasques són importants i no hi ha gaire espai per retallar funcionalitats addicionals, el que

implica que la feina s'ha de realitzar setmanalment i de forma disciplinada, constant i rigorosa. D'aquí la necessitat de provar les parts de codi implementades tan aviat com sigui possible i fer un molt bon disseny i planificació de les parts a implementar: si els errors inesperats es detecten ràpidament, es poden dedicar i distribuir hores addicionals per al projecte; trobar els errors massa tard, en canvi, podria resultar en pèrdues de temps molt més difícils de compensar. Tot i això, cal dir que l'assignació de temps a cada part de la implementació del projecte ja contempla el temps que s'hauria de dedicar a aquest tipus de problemes, sempre i quan es localitzin prou aviat.

3. *Pressupost i sostenibilitat*

3.1 Gestió econòmica

En aquest apartat es descriuen els costs econòmics dels recursos humans, hardware, software i d'altres despeses indirectes implicats en el projecte, així com els pressupostos corresponents.

Software

A nivell de software els costs previstos són nuls. Això és així ja que tot el projecte es basa en software lliure, i els sistemes operatius que s'utilitzaran durant el projecte són sistemes operatius basats en Linux. La vida útil del software no s'inclou ja que totes les actualitzacions previstes també són gratuïtes, donades les llicències o característiques del software.

Software	**Preu**	**Amortització**
FreeLing	0.0€	0.0€
Word2Vec	0.0€	0.0€
API de Twitter	0.0€	0.0€
Sistemes Operatius	0.0€	0.0€
Total	**0.0€**	**0.0€**

Recursos Humans

A nivell de recursos humans, les tasques realitzades seran de diferents tipus. A continuació podem veure el pressupost esperat:

Rol	Hores	Preu per hora	Cost total
Cap de projecte	100h	50.0€/h	5000.0€
Analista	170h	35.0€/h	5950.0€
Programador	180h	25.0€/h	4500.0€
Tester	50h	25.0€/h	1250.0€
Cost Tasques	**500h**	—	**16700.0€**
Seguretat social	—	—	**5511.0€**
Total	—	—	**22211.0€**

Els sous s'han obtingut prenent com a referència ofertes de feina reals, mentre que les hores s'han calculat en base a la planificació i el diagrama de Gantt prèviament obtinguts. El cost de la seguretat social s'ha calculat com un 33% sobre la suma del cost de les tasques. La major part d'hores de treball del cap de projecte s'extreuen de l'etapa de planificació del projecte. Sobre els rols d'analista i programador, les hores realitzades es corresponen al gruix del projecte i les diverses etapes d'implementació. En gairebé totes elles es requereixen els dos rols, tot i que en diferents graus. Finalment, les hores de treball del tester o provador de software es corresponen a les proves que cal realitzar després de cada etapa d'implementació. En especial, la feina es concentra al termini de les implementacions i entrenament de cadascun dels models de llenguatge implicats en el projecte.

Hardware

L'únic hardware necessari per al projecte, en principi, són els ordinadors de treball que s'utilitzaran. Per als càlculs de l'amortització s'han considerat 1780 hores hàbils anuals.

Hardware	Preu	Vida útil	Amortització
Ordinador sobretaula	700.0€	5 anys	23.3€
Portàtil	450.0€	4 anys	18.7€
Total	**1150.0€**	—	**42.0€**

Donat que aquest hardware no ha estat específicament comprat per al projecte, els costs d'amortització es consideraran costs indirectes.

Altres despeses indirectes

En aquesta secció es mostren altres despeses indirectes del projecte, com és el consum d'electricitat. Aquest consum s'ha calculat a partir dels diferents consums de l'ordinador de sobretaula i el portàtil. Els resultats obtinguts han estat de 60KWh per l'ordinador de sobretaula i 37KWh pel portàtil, un total de 97KWh consumits.

Producte	Preu unitat	Unitats	Cost total
Electricitat	0.12€/KWh	97KWh	11.64€

Els costs mediambientals es discuteixen més endavant a la secció de sostenibilitat.

Contingència

La partida de contingència serà la partida del pressupost destinada als possibles errors o desviacions donats en el pressupost. Això considera exclusivament els costs directes i indirectes pressupostats, i es calcula com el 15% de la seva suma. El valor final de la partida de contingència es detalla en el pressupost final.

Imprevistos

Tal com ja es va explicar a la planificació, un dels major perills del treball són les desviacions temporals respecte la planificació. En cas que es donessin aquestes desviacions, això tindria una influencia directa en els costs directes detallats fins ara. Alhora, aquest increment dels costs directes tindria repercussions sobre els costs indirectes, que també depenen del volum de treball realitzat. Així doncs, s'afegirà una partida d'imprevistos en la implementació del software al pressupost final. El valor d'aquesta partida està calculat a partir d'un 5% de risc d'un increment en les hores dedicades a la implementació del projecte, amb els costos directes i indirectes associats.

Pressupost Total

En aquesta secció es recopilen els pressupostos detallats a les seccions prèvies i es mostra el pressupost complet:

Activitat	Cost
Recursos Humans	22211.0€
Software	0.0€
Total costs directes	**22211.0€**
Amortització Hardware	42.0€
Electricitat	11.64€
Total costs indirectes	**53.64€**
Contingència	**3339.696€**
Imprevistos	**111.05€**
Total (sense IVA)	**25715.386€**
Total (amb IVA)	**29829.8478€**

Control de gestió

En tot projecte de software, en moltes ocasions estimar la complexitat dels problemes a resoldre pot ser molt complicat. Tot i que els imprevistos ja es tracten de tenir en compte als pressupostos, la desviació en els costs sorgits per aquest tipus de problemes pot arribar a ser elevada. Així doncs, encara que les probabilitats de que això succeeixi siguin reduïdes, és un punt que cal tenir molt en compte. En el cas de les desviacions petites, aquestes no haurien de ser massa problema ja que es poden compensar dins el pressupost i terminis establerts.

En el cas de desviacions més importants, s'aplicaria el definit en la planificació del projecte: d'una banda s'haurien d'incrementar les hores dedicades al projecte en les setmanes posteriors a les desviacions. Això òbviament comportaria un increment en els costs directes dedicats als recursos humans. L'altra possibilitat detallada en la secció de planificació del projecte és la de retallar certes funcionalitats en l'etapa d'implementació. Això no impactaria el cost del projecte negativament, però sí tindria potencialment un efecte negatiu sobre la qualitat del resultat final, cosa que no seria desitjable.

Finalment, en cas que es donin desviacions que podrien haver estat previstes amb una millor planificació, aquestes s'anotaran i analitzaran amb l'objectiu que no tornin a succeir o es puguin tenir en compte en possibles projectes posteriors.

3.2 Lleis i regulacions

A continuació es tracten les lleis, regulacions i llicències que apliquen a aquest projecte i als recursos que s'utilitzen.

D'una banda hi ha les llicències de software lliure que afecten a les dues principals eines utilitzades durant el projecte, FreeLing i Word2Vec.

FreeLing està llicenciat sota la *GNU Affero General Public License*[1], una llicència que permet la modificació del codi sempre que es compleixin una sèrie de condicions: si els programes derivats es distribueixen públicament cal fer el codi font públic sota una llicència compatible, incloure la llicència amb aquest, especificar les condicions d'ús, llicències i copyright originals i explicitar els canvis significants realitzats al codi.

Word2Vec en canvi està llicenciat sota la *Apache License 2.0*[2]. Respecte a la modificació del codi, aquesta llicència és molt similar i compatible amb la llicència de FreeLing.

[1] `http://www.gnu.org/licenses/agpl.html`
[2] `http://www.apache.org/licenses/LICENSE-2.0`

D'altra banda hi ha Twitter. L'API de Twitter és usada puntualment en el projecte per extreure tweets i fer proves amb el mòdul de correcció. Tal com es pot llegir a les normatives legals de Twitter[3], hi ha alguns punts importants que poden afectar al projecte, com la necessitat d'eliminar i/o modificar el contingut extret de Twitter quan així sigui requerit pel servei. En el projecte l'API de Twitter no s'utilitza per a cap aplicació pública, sinó únicament per a realitzar proves i avaluar el funcionament del mòdul de correcció de forma privada. Aquest procés requereix emmagatzemar tweets de forma temporal per a poder-los processar amb el corrector, però un cop processats es tornen a eliminar, de forma que no s'incompleix la normativa de Twitter. Cal tenir en compte que en cas d'incomplir-la no només s'estarien violant els termes d'ús de Twitter, sinó que potencialment també es podria incórrer en una violació de la LOPD (Ley Orgánica de Protección de Datos de Carácter Personal). En els casos en que cal mantenir un conjunt de tweets per a proves o demostracions off-line, els continguts s'anonimitzen de forma que no puguin comprometre als usuaris originals.

3.3 Sostenibilitat

Sostenibilitat Econòmica

Els costs del projecte han estat estimats de la forma el més realista possible, tenint en compte els costs directes, indirectes, possibles imprevistos i impostos aplicats.

En aquest cas, al tractar-se d'un projecte de codi lliure, és complicat valorar la competitivitat econòmica del projecte. D'una banda, els costs pressupostats són bàsicament de recursos humans, que són indispensables per a la tasca, i la resta de costs no són particularment elevats ni senzilla o legalment evitables. Tot i això, l'objectiu del projecte no és crear un producte que pugui generar ingressos que compensin els costs necessaris per crear-lo. Si el projecte suposa eventualment beneficis econòmics per a algú, serà de forma indirecta.

El temps dedicat a cada apartat del projecte, a més, és l'apropiat. Hi ha diverses tasques importants de similar complexitat, i totes elles tenen un temps de desenvolupament raonable assignat. Per tant, no s'està retallant ni exagerant en la planificació temporal de forma que es redueixin o augmentin els costs del projecte. Si ho féssim, aleshores la qualitat del producte resultant es veuria molt probablement afectada: en cas de dedicar menys temps, la qualitat del software desenvolupat el faria difícilment mantenible o fiable; en cas de dedicar-ne més, tot i que la qualitat del software podria millorar, molt probablement no ho faria de forma significativa o eficient respecte al temps addicional empleat per a aconseguir-ho.

Respecte al manteniment del projecte, en cas que finalment els models de llen-

[3] `https://dev.twitter.com/overview/terms/agreement-and-policy`

guatge implementats s'incorporessin a FreeLing, el projecte passaria a formar part d'un altre projecte major i que compta amb diversos grups que s'encarreguen del seu manteniment i desenvolupament, com pot ser el TALP[4]. Cal dir que els models com a tal no requereixen gaire manteniment a priori, però la potencial implementació de noves tècniques que en milloressin l'eficiència seria perfectament realitzable amb costs raonables, donada la correcta documentació del codi desenvolupat.

Sostenibilitat Social

La necessitat i els efectes positius de la tecnologia són qüestionats molt sovint. En el cas del processament de llenguatge natural, les aplicacions possibles són moltes i molt diverses. D'una banda, les millores en PLN poden ser considerades positives, ja que entre d'altres posen a disposició de molta gent traductors automàtics, correctors, assistents personals intel·ligents, etc. D'altra banda, es pot considerar que això també tindrà efectes negatius, i que a llarg plaç es perdran molts llocs de treball (fins i tot actualment ja podem veure contestadors automàtics robotitzats).

Aquest projecte només és un granet de sorra dins aquest gran escenari, i socialment no té cap importància directa significativa, a excepció de la que pugui suposar per als desenvolupadors i usuaris de FreeLing. En aquest aspecte, el projecte sí podria ser útil, ja que podria posar els word embeddings a disposició de desenvolupadors d'aplicacions que requereixin PLN, a més de poder servir com a base per a futurs estudiants o investigadors que vulguin fer recerca sobre aquest tema en particular.

Molt probablement el desenvolupament d'aquestes tecnologies no serà un fet exclusivament positiu per a la qualitat de vida de les persones. És raonable pensar que, tal com s'ha suggerit, l'ús d'aquestes tecnologies també comporti certs problemes a nivell social. Tot i això, aquest desenvolupament de la tecnologia és inevitable, i en última instància la tecnologia té un gran potencial per facilitar i millorar la qualitat de vida de les persones. A la pregunta de si aquest potencial s'acabarà realitzant, això ja dependrà de la forma en que s'acabi usant la tecnologia que desenvolupem.

Tenint en compte aquest escenari, és molt complicat predir el que acabarà succeint. Aquest projecte realment només podrà tenir un impacte social important de forma indirecta si d'altres productes o aplicacions futures n'utilitzen els resultats. I donat aquest cas, considero que el fet de posar a disposició el software com a software lliure és una bona direcció a prendre socialment, ja que posa el coneixement i les possibilitats d'aprofitar-lo a disposició de qualsevol.

[4] `http://www.talp.upc.edu/`

Sostenibilitat Mediambiental

L'impacte del projecte a nivell mediambiental és de l'ordre del mínim possible per a un projecte d'aquestes característiques. El hardware utilitzat és el mateix que faig servir per a treballar usualment, i el projecte simplement resultaria impossible de realitzar sense hardware. El processament de llenguatge natural es basa precisament en l'ús dels computadors, i com a tal no es pot realitzar de la mateixa forma sense ells.

És cert que aquest hardware té un impacte mediambiental no desitjable, tant a nivell d'emissions de CO_2 com a nivell dels components i materials o condicions de treball requerits per a fabricar-los, com pot ser particularment el cas del coltan. Malgrat tot, considero que l'ús de la computació és un fet positiu a nivell tecnològic que d'una banda ens permet dedicar menys temps a tasques repetitives i mecàniques, i d'altre banda ens ofereix una infinitat de possibilitats, entre les quals s'inclouen les de millorar l'eficiència i reduir els costs mediambientals dels productes que utilitzem diàriament. De fet, sense els computadors no podríem ni tan sols avaluar de forma realista els costs mediambientals que tenen les nostres accions en la societat en la que vivim actualment.

Per a aquest projecte ni els costs ni els beneficis mediambientals són particularment elevats, i podem concloure que tampoc existeix cap forma òbvia de canviar aquest impacte sense canviar les especificacions i objectius del projecte.

Finalment, podem veure una estimació dels costs de CO_2 per a aquest projecte. Aquests costs es deriven de l'ús del hardware que es preveu utilitzar:

- **Cost de fabricació**: aproximadament 600kg de CO_2 entre els dos ordinadors. Amortitzats al temps d'ús del projecte, suposen 25kg d'un ordinador i 20kg de l'altre, sumant un total de 45kg.

- **Cost d'ús**: a partir de les aproximacions realitzades a l'apartat de costs indirectes i basant-nos en el consum d'electricitat, el cost d'ús en kg de CO_2 per als ordinadors arriba a un total de 63.05kg de CO_2.

En total, el cost aproximat en kilograms de CO_2 per al desenvolupament del projecte és de 108.05kg.

Matriu de sostenibilitat

En base a les consideracions dels apartats previs s'ha elaborat la següent matriu de sostenibilitat:

	PPP	**Vida útil**	**Riscos**
Ambiental	Consum del disseny 7	Petjada ecològica 9	Riscos ambientals 0
Econòmic	Factura 4	Viabilitat 8	Riscos econòmics -2
Social	Impacte personal 5	Impacte social 7	Riscos socials -1

Els valors estan escalats tal com es detalla a l'article original que proposava la matriu de sostenibilitat [21], amb la primera columna de 0 a 10, la segona de 0 a 20 i l'última de -20 a 0. El total obtingut és de 37, el que indica que aquest és un projecte sostenible, sense riscos importants ni costs ambientals significatius, que a més es viable econòmicament i pot tenir cert impacte social positiu.

4. *Revisió de la planificació*

En aquest capítol s'enumeren els canvis i desviacions respecte la planificació original del projecte, així com els motius d'aquestes desviacions i la forma en que s'han resolt.

4.1 Canvis en la metodologia

La metodologia de treball no ha patit cap canvi significant. S'han mantingut reunions setmanals amb el director del projecte i el codi s'ha anat millorant de forma incremental. El ritme de desenvolupament del codi ha estat raonable i no ha calgut considerar metodologies alternatives, tot i que sí hi ha hagut alguns canvis en la planificació del projecte.

4.2 Canvis en la planificació

A continuació es presenta la taula temporal amb les tasques de la planificació original del projecte.

Tasca	**Duració aproximada**
Planificació del projecte	80
Configuració i familiarització amb l'entorn de treball	20
Integració de word embeddings a FreeLing	100
Obtenció de dades i entrenament del model	60
Implementació del mòdul de normalització	80
Implementació dels models a comparar	90
Anàlisi dels resultats	20
Tasca final	50
Total	**500**

Des d'aquesta planificació original, el projecte ha patit alguns canvis per diversos motius.

En primer lloc, els algoritmes que s'havia considerat utilitzar per a trobar la combinació òptima de correccions han resultat no ser adequats. En un primer moment s'havia considerat utilitzar l'algoritme de Viterbi, un algoritme de programació dinàmica capaç de trobar la seqüència més probable (o camí òptim) d'un espai d'estats determinat. Aquest algoritme pot funcionar correctament amb altres models de llenguatge com els 3-grames, però no és adequat pels word embeddings. En el seu lloc s'ha optat per utilitzar mètodes d'optimització més generals, com els algoritmes genètics.

En segon lloc, després de treballar amb els word embeddings i avaluar els primers resultats obtinguts, s'han detectat certs problemes i s'ha vist que els word embeddings, a diferència per exemple dels 3-grames, tenen més problemes per a donar bons resultats si s'utilitzen de forma aïllada, i cal incloure també altres mètriques durant el procés de normalització. La causa de tot això és que tot i que diversos models de llenguatge poden utilitzar-se per a resoldre els mateixos problemes, la informació que aporten pot tenir propietats molt diferents. Això implica que calen aproximacions potencialment molt diferents per a treballar amb els diversos models. Durant el transcurs del projecte s'han anat veient totes aquestes diferències i tractant de resoldre els nous problemes que sorgien.

Totes aquestes situacions òbviament han portat a fer algunes modificacions a la planificació. A nivell del temps de treball dedicat, aquest s'ha mantingut similar, però ha calgut redistribuir-lo. En particular, ha calgut dedicar més temps a la implementació del mòdul de normalització en sí mateix i a experimentar i analitzar amb més detall les propietats dels word embeddings. A canvi d'això, ha calgut deixar de costat la implementació pròpia dels n-grames, i realitzar les comparacions dels resultats exclusivament amb els del Tweet-Norm del 2013.

La planificació actualitzada es mostra a continuació:

Tasca	**Duració aproximada**
Planificació del projecte	80
Configuració i familiarització amb l'entorn de treball	20
Integració de word embeddings a FreeLing	80
Obtenció de dades i entrenament del model	70
Implementació del mòdul de normalització	150
Anàlisi dels resultats i comparació de models	50
Tasca final	50
Total	**500**

Respecte el pressupost del projecte, aquest s'ha mantingut igual. Bàsicament s'han redistribuït les hores a dedicar a cada part del projecte, però el total d'hores s'ha mantingut, i no han sorgit costos addicionals ni pel hardware a utilitzar ni pels recursos humans.

Finalment, cal destacar que tot i que els objectius del projecte s'han mantingut els mateixos, sí s'ha desplaçat lleugerament el focus en alguns punts. En particular, quan s'ha vist que elaborar un mòdul de correcció utilitzant word embeddings no era tan directe com intercanviar un model de llenguatge per un altre, s'ha decidit dedicar més atenció a l'estudi de les propietats i característiques úniques d'aquests.

5. *Estructura d'un corrector automàtic*

En aquest capítol es descriuen l'estructura bàsica i parts principals d'un corrector automàtic. L'objectiu es donar una visió general i clara de com funciona el procés de correcció, que permeti seguir més fàcilment els capítols següents en els que s'entra en detalls més avançats sobre la implementació del projecte.

5.1 Pipeline bàsic

El procés bàsic de correcció es mostra en l'exemple de la figura 5.1. En primer lloc es segmenta la frase en diferents parts, usualment fent servir els espais de la frase per separar cada paraula. A continuació, es busca quines de les paraules són incorrectes. Després, es generen alternatives per a aquestes paraules incorrectes, i finalment es fa la selecció de les millors alternatives.

De manera més formal, podem dividir el procés en les tasques de *tokenització*, detecció de paraules incorrectes, generació d'alternatives i selector d'alternatives, tal i com es mostra a la figura 5.2. Aquest és l'esquema bàsic utilitzat en un corrector, i és el que s'usa en aquest projecte.

5.2 Descripció de les parts

En aquesta secció s'analitzen amb més detall les parts del procés de correcció prèviament mencionades i s'indica com es poden realitzar algunes de les tasques utilitzant FreeLing.

Tokenitzador

El tokenitzador és l'encarregat de dividir la frase o text en fragments que anomenem *tokens*. En la majoria dels casos, els tokens no són més que paraules, i la tokenit-

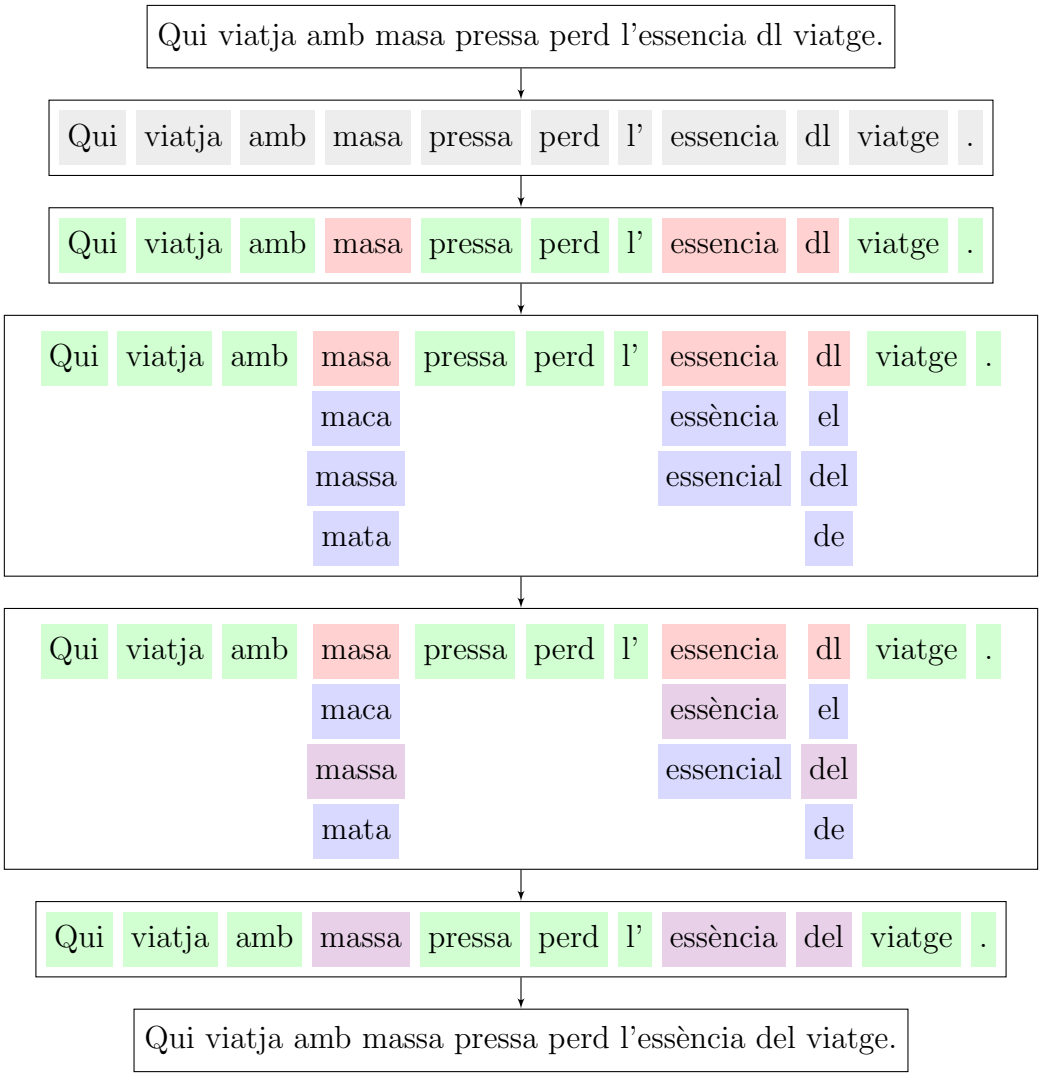

Figura 5.1: Exemple complet del procés de correcció. La frase es tokenitza, es detecten les paraules incorrectes, es generen alternatives i finalment es seleccionen les millors. Text basat en una cita de Louis L'Amour.

zació d'una frase pot ser tan senzilla com dividir les paraules i separar els signes de puntuació que trobem units a algunes d'elles.

Alguns tokenitzadors consideren els espais i salts de línia com a tokens, mentre que d'altres els ignoren completament. En general aquest tipus de caràcters no tenen major importància i utilitat que la de reconstruir el text original un cop analitzat o modificat pertinentment. La puntuació en canvi sí s'acostuma a mantenir.

Tot i que el procés sembli inicialment senzill, a l'hora d'aplicar-lo al llenguatge natural poden aparèixer diverses complicacions, incloent problemes d'ambigüitat. A més en el cas d'aquest projecte estem tractant directament amb text que pot estar escrit incorrectament, i per tant el tokenitzador ha de ser capaç de respondre a aquestes situacions.

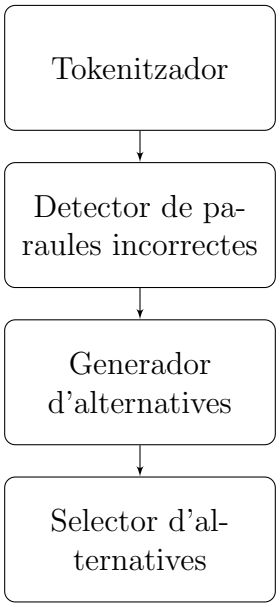

Figura 5.2: Pipeline bàsic per al procés de correcció automàtica.

Un dels problemes es la puntuació. Tal com ja s'ha mencionat, en molts casos caldrà separar les paraules de puntuació unida a elles. Però és fàcil veure que hi ha altres expressions habituals on la puntuació s'usa de formes molt diferents:

- Nombres que utilitzen punts, comes, espais i apòstrofs per separar milers, decimals, seccions d'un text, etc. Un altre cas interessant són per exemple els nombres en notació científica. Finalment també cal tenir en compte que en alguns idiomes són vàlides expressions com "*l'11 de setembre*" o d'altres expressions on es combinen lletres i nombres de formes complexes.
- Emoticones, que poden contenir combinacions de tot tipus de símbols de puntuació amb d'altres lletres, com poden ser: "`:p`", "`;-D`", "`o_o`", "`=)`", "`\o/`", etc.
- URLs, que es poden confondre amb finals i inicis de frase que no han estat correctament separats. En el passat gairebé sempre s'utilitzava el prefix "*www.*" i un conjunt d'extensions de domini bastant limitat, però avui dia els noms de les pàgines web poden aparèixer sense cap prefix i existeixen centenars d'extensions de domini.
- Abreviatures: "*adv.*", "*att.*", "*c/*", "*st.*", etc.

Com es pot veure hi ha una gran diversitat de casos en els que cal fer un tractament més refinat del text d'entrada. Un altre exemple podria ser la contracció dels pronoms en moltes llengües, que normalment convé separar per al posterior anàlisi.

Un altre tema són els compostos. En ocasions hi ha paraules que apareixen freqüentment juntes i que tenen un significat diferent al que tenen els mots per

separat. Un exemple senzill i que s'utilitzarà també més endavant és "*xarxes socials*". A l'hora d'analitzar un text, si trobem aquesta expressió gairebé sempre ens interessarà considerar-la com a una única unitat. Aquests compostos s'acostumen a indicar fent servir un guió baix: "*xarxes_socials*".

Tècnicament la detecció dels compostos, en cas que es realitzi, es pot fer en una part del procés posterior a la tokenització. Tot i així, fer-ho durant la tokenització usualment té sentit ja que és el moment en que es construeixen els tokens.

Com es pot veure, tot i que un procés de tokenització pugui ser molt senzill en la seva forma més bàsica, també pot arribar a ser molt complex, en funció de per a què s'utilitzi i com de precís necessitem que sigui.

Tokenitzador a FreeLing

Tal com es pot esperar, FreeLing compta amb el seu propi tokenitzador, el que facilita molt aquesta tasca. A més, FreeLing permet l'ús de fitxers de configuració per personalitzar encara més el funcionament del tokenitzador i reconèixer certs patrons o casos a partir d'expressions regulars.

Per al cas d'aquest projecte, per exemple, s'utilitzen fitxers de configuració preparats per a textos provinents de Twitter, que poden incloure emoticones i d'altres expressions pròpies del llenguatge escrit informal o llenguatge xat.

Detector de paraules incorrectes

El següent pas en el procés de correcció és la detecció de paraules incorrectes. Aquest pas habitualment es resol comparant les paraules del text amb un diccionari.

En aquest projecte les úniques paraules que es corregeixen són aquelles ortogràficament incorrectes, però tècnicament és possible anar més enllà i considerar una paraula incorrecta no només per qüestions ortogràfiques sinó també morfològiques. Per exemple, la frase "*El diner no donen la felicitats*", tot i ser ortogràficament correcta, no està formalment ben construïda.

Si deixem de banda aquest tipus d'errors i considerem únicament els errors ortogràfics, la detecció de paraules incorrectes es relativament senzilla. La part més complicada és la de l'elaboració dels diccionaris o llista de paraules acceptades, però avui en dia existeixen molts diccionaris i corpus de text d'on es pot extreure aquesta informació. Per a aquest projecte ni tan sols cal implementar aquesta part, ja que FreeLing ja compta amb un analitzador que fa aquesta feina.

Tot i això, cal tenir en compte alguns aspectes més. En primer lloc, hi ha idiomes en els que aquesta tasca resulta més complicada que en d'altres. En castellà i català, per exemple, les formes verbals varien en funció de la persona, el nombre, i el temps

verbal. Existeixen a més moltes conjugacions irregulars que fan encara més difícil la feina de saber si un verb és o no correcte. En aquest aspecte el català i el castellà presenten més dificultat que d'altres idiomes com l'anglès. En alemany per exemple trobem la composició de substantius, que també cal tenir en compte. Com es pot veure, tots els idiomes tenen les seves particularitats, i es per això que la dificultat d'aquesta tasca de detecció de paraules incorrectes també pot variar d'idioma a idioma.

Un altre fenomen important és el dels estrangerismes o la citació o ús deliberat de paraules d'un altre idioma. En aquests casos acostuma a ser difícil decidir si una paraula és incorrecta o no, o si existeix alguna traducció raonable. Un cas similar són els noms d'empreses, o noms propis que en un text poden aparèixer incorrectament escrits en minúscules.

Les llengües a més no són estàtiques, sinó que estan en continua evolució i procés de canvi. Tenir en compte únicament les paraules reconegudes per diccionaris oficials por tenir més o menys sentit en textos formals, però a l'hora de tractar textos informals aquest procés de decidir què és correcte i què no pot generar molts dubtes. En l'apartat de la implementació del corrector s'explica en més detall com s'ha tractat de resoldre aquest problema en el projecte.

Generador d'alternatives

Un cop tenim les paraules incorrectes cal generar alternatives correctes per a totes elles. Per a aquest procés usualment s'utilitza el concepte de distància d'edició.

La distància d'edició és el nombre de transformacions que s'ha d'aplicar a una paraula per a que es converteixi en una altra. Així, si tenim la paraula "*escriu*", fent una única transformació podem obtenir d'altres paraules com "*descriu*" o "*escrit*". "*Espriu*" estaria a una o dues transformacions de distància, en funció de si comptem el canvi a majúscula de la e o no. Amb dues transformacions aconseguim paraules com "*estiu*", "*estris*", "*estrip*". La paraula "*cridar*", en canvi, està a una distància d'edició de cinc.

En aquest cas s'ha considerat com a transformacions vàlides:

- Eliminar una lletra a una paraula.
- Afegir una lletra a una paraula.
- Intercanviar una lletra per una altra.

Aquesta distància d'edició sovint s'anomena distància de Levenshtein, en referencia al seu autor original. Tot i això, aquesta mètrica és molt senzilla, i usualment s'utilitzen algoritmes lleugerament més sofisticats com el de Damerau-Levenshtein,

de l'any 1964, en que també es considera la transposició de lletres [3]. Utilitzant aquest algoritme, per exemple, es considera que les paraules "*casual*" i "*causal*" estan a distància d'edició u.

En base a aquestes distàncies es poden buscar en un diccionari paraules properes a aquelles que no s'hagin reconegut en un text. Tot i això generar alternatives d'aquesta manera no és l'única possibilitat, i es poden utilitzar moltes mètriques i sistemes més sofisticats. Per exemple, els guanyadors del Tweet-Norm del 2013 van utilitzar un sistema basat en autòmats finits que va donar resultats lleugerament millors que la distància de Levenshtein clàssica [18].

Generador d'alternatives a FreeLing

FreeLing ja compta amb un mòdul de generació d'alternatives que funciona basant-se en les idees explicades de la distància d'edició. A més, FreeLing compta amb algunes millores addicionals com la possibilitat de tenir en compte la fonètica de les paraules a l'hora de calcular les seves distàncies d'edició. Així, paraules com "*amable*" i "*amapla*" es consideren molt similars, tot i que gràficament canviïn dues lletres.

De fet, les distàncies que utilitza FreeLing no són tan simples com el nombre de transformacions realitzades, sinó que cada tipus de transformació té un pes o un cost específics. A més, aquestes transformacions i costos es poden editar utilitzant fitxers de configuració, el que permet afinar més el procés en cas de necessitat.

Tot i això, tal com s'explicarà en seccions d'implementació posteriors, durant el projecte ha calgut considerar alguns elements més a l'hora de treballar amb les distàncies d'edició, ja que els resultats inicials que s'obtenien de FreeLing no eren prou satisfactoris donats els requeriments del projecte i les tècniques utilitzades.

Selector d'alternatives

La selecció de l'alternativa més adequada és una de les possibles parts finals del procés de correcció automàtica. Un cop més, aquest procés pot variar des de ser molt simple fins a ser extremadament complex.

Aquesta part del procés és en la que més s'enfoca aquest projecte, ja que es tracta d'utilitzar els word embeddings com a alternativa a mètodes més tradicionals. Però tractem els mètodes més senzills primer.

El mètode més senzill per solucionar el problema és utilitzar les pròpies distàncies d'edició que ja podem tenir del procés de generació d'alternatives per ordenar les solucions. De fet, si utilitzem únicament el nombre de transformacions, possiblement

tindrem moltes alternatives amb cost u (una única transformació), i per tant també voldrem ordenar els resultats lexicogràficament.

Tot i això, tal com es veurà en seccions posteriors, aquesta opció no dóna massa bons resultats. Fins i tot quan no usem mètriques de distàncies tant simples i utilitzem estratègies com les explicades anteriorment de donar diferents pesos a diferents transformacions, els resultats encara deixen molt que desitjar. Malgrat tot, aquesta pot ser una opció apropiada per a editors de text que permetin a l'usuari seleccionar la millor correcció d'entre les possibilitats donades.

3-Grames

Tradicionalment una de les estratègies més utilitzades i que millors resultats ofereix són els 3-grames. Els 3-grames, tal com s'ha comentat en els capítols inicials de la memòria, són una de les possibles formes dels n-grames.

Els n-grames són un model de llenguatge que assigna una probabilitat a una seqüència donada de N paraules. En el cas dels 3-grames, aquestes seqüències són de tres paraules.

El càlcul d'aquestes probabilitats es realitza en base al nombre d'aparicions de la seqüència en un corpus de text. Per exemple, si prenem la seqüència en castellà "*tres tristes tigres*" resulta que aquesta té una probabilitat de 1.2641×10^{-6}. És a dir, que de cada seqüència de tres paraules que podríem trobar en un text, un 0.00012641% de les vegades aquesta seqüència és "*tres tristes tigres*".

De el paràgraf anterior no és totalment correcte, i enlloc d'"un text" seria més apropiat dir "el corpus de text utilitzat per a l'entrenament del model utilitzat". Quan el corpus de text utilitzat és prou gran, els resultats es poden extrapolar cada cop de forma més fiable, a excepció dels casos en que es tracten dominis particulars i pot ser convenient entrenar els models amb corpus més específics.

Aquesta probabilitat de l'exemple, en particular, s'ha obtingut usant una eina en línia que ofereix Google anomenada Google Books Ngram Viewer [11], accessible a `https://books.google.com/ngrams`. El resultat donat es pot veure gràficament a la figura 5.3.

Aquesta figura també és molt interessant ja que ens mostra l'estat de continu canvi de la llengua del qual ja s'ha parlat anteriorment. En aquest exemple podem veure que el popular embarbussament probablement va aparèixer al voltant del 1934, i el seu ús per escrit s'ha incrementat dràsticament en les últimes dècades. De fet aquest no és ni tan sols un cas extrem, i podríem veure com amb termes més recents, per exemple aquells relacionats amb la tecnologia, les probabilitats dels 3-grames associats encara canvien més bruscament en terminis de temps molt breus.

Més enllà encara, si reflexionem sobre termes i expressions que formen part de

modes passatgeres o continguts virals a internet, ens adonem que efectivament és molt complicat tenir models que realment reflecteixin la llengua que s'utilitza en un moment concret de la història.

Figura 5.3: Resultat de cercar el 3-grama "*tres tristes tigres*" (en minúscules, tot i que de fet l'expressió és més habitual amb majúscules) a Google Books Ngram Viewer.

Òbviament, també hi ha d'altres factors que podrien ser significants, com el lloc d'origen d'un text, o el context del seu autor. En conclusió, la complexitat de les situacions en les que ens trobem en el processament de llenguatge natural pot arribar a ser molt elevada.

Tornant als n-grames, la idea fonamental que fa que funcionin bé per a la tasca de correcció és que son capaços d'identificar el context i l'ordre en el que tendeixen a aparèixer les paraules, i d'aquesta manera donen molta informació sobre les seqüències de paraules que potencialment tenen més sentit. Per exemple, si prenguéssim els 2-grames "*redes sociales*" i "*redes rurales*", resulta que el primer té una probabilitat molt major, de 0.0001560605, davant la probabilitat de 0.0000005216 de "*redes rurales*". Així, veiem que "*redes sociales*" és gairebé tres-centes vegades més probable que "*redes rurales*".

Per tant, si en un text trobéssim una expressió com "*redes suriales*", on el terme incorrecte "*suriales*" resulta tenir una distància d'edició de Levenshtein dos tant amb el terme "*sociales*" com "*rurales*" (i per tant les dues podrien haver estat suggerides com a possibles alternatives), gràcies als n-grames probablement ens decantaríem per "*redes sociales*" com a alternativa més probable.

Aquest model probabilístic funciona bé en la majoria de casos, i els 3-grames en particular són un dels tipus de n-grames que millors resultats dóna en aquesta tasca, usualment amb precisions superiors al 75% [18].

Un altre factor a tenir en compte dels n-grames és que són models que requereixen grans quantitats de memòria. És fàcil pensar que la combinatòria de possibles seqüències de N paraules explota ràpidament. Això té algunes conseqüències negatives, com la necessitat d'emmagatzemar els models a disc dur al no caber en

memòria RAM, o la necessitat d'utilitzar tècniques d'indexació per accedir a les dades de forma prou eficient. A la taula 5.1 es poden veure les mides per a diferents models d'n-grames entrenats per Google en anglès [4].

Model	Mida
2-grames	7,77 GB
3-grames	35,22 GB
4-grames	69,06 GB
5-grames	80,62 GB

Taula 5.1: Mides de diferents models d'n-grames.

Tal com es pot inferir de la taula hi ha algunes tècniques per reduir la mida dels models, i existeixen un gran nombre de combinacions de paraules que no es donen mai i no s'inclouen al model.

A més, en el cas de les seqüències que apareixen molt poc freqüentment, els resultats poden ser poc precisos. La distribució dels n-grames segueix la llei de Zipf: la majoria de seqüències són infreqüents, mentre que una minoria apareix de forma molt habitual. En altres paraules, hi ha una gran quantitat de seqüències amb baixa freqüència d'aparició en que l'error comés a l'estimar la seva probabilitat a partir d'un corpus pot ser significant. Per resoldre aquest problema existeixen les anomenades tècniques d'*smoothing* o suavitzat, com poden ser l'*smoothing* de Kneser-Ney o alguna de les seves variacions posteriors, que tracten d'ajustar les probabilitats experimentals amb la distribució probabilística esperada [15].

Selecció d'alternatives a FreeLing

FreeLing no compta amb cap mètode de selecció d'alternatives més que la informació que es calcula sobre la distància d'edició de les paraules. Aquest és el tema en el que es centra el treball, i no s'utilitza cap de les estratègies explicades fins al moment, sinó que s'utilitzen els word embeddings, un model de llenguatge molt diferent als 3-grames. Els detalls sobre els word embeddings com a model es tracten al capítol 6.

5.3 Complexitat del problema

Fins ara s'ha presentat l'esquema més bàsic de correcció automàtica, però és possible enfocar el problema de forma diferent o afegir alguns passos en el procés. En alguns casos es pot intentar fer diversos anàlisis just després de la tokenització o la detecció de paraules incorrectes, com poden ser anàlisis sintàctics o d'altres tipus. El problema en aquests casos es que si hi ha paraules incorrectes al text, l'anàlisi ha de poder treballar amb informació incompleta o incorrecta. Això no acostuma

a passar, i es per això que la correcció resulta un problema tan complicat: és molt difícil decidir computacionalment si una frase té sentit o no, encara més quan no està aïllada i hem de considerar-la en un context més ampli.

La forma en la que els humans corregeixen mentalment les frases, en canvi, sí que aprofita aquesta idea. Ràpidament som capaços de veure i decidir si una paraula és correcta o incorrecta, si una frase té sentit o no, buscant alternatives a les paraules errònies i re-avaluant la frase de forma immediata.

El que cal veure aquí és la diferència entre el pensament humà i les tècniques aplicables computacionalment avui en dia. Les persones som capaces de treballar amb informació incompleta de forma molt natural, i no tenim problemes per analitzar i explorar aquestes frases incorrectes. En canvi, els mètodes computacionals actuals de PLN són molt menys flexibles, i no tenen una noció real sobre si les frases tenen més o menys sentit, ja que no són capaços d'abstraure el significat o entendre les frases.

Al reflexionar sobre aquest punt podem veure la diferència i la distància entre la forma en que les persones i els ordinadors poden enfocar el problema, així com les dificultats i limitacions que implica no poder-lo tractar de la mateixa manera.

6. *Word embeddings*

En aquest capítol es tracten en major profunditat els word embeddings, s'explica el seu funcionament i les seves propietats rellevants per a la tasca de correcció.

6.1 Models usats en l'entrenament

Els word embeddings, a diferència dels n-grames, utilitzen tècniques d'aprenentatge màquina per a aconseguir representacions de les paraules d'un corpus. Més concretament, utilitzen xarxes neuronals i extreuen la representació d'una paraula de la capa oculta de la xarxa neuronal un cop aquesta ha estat entrenada.

La xarxa utilitzada i la funció objectiu a maximitzar depenen del model utilitzat [20]. Els models utilitzats per a l'entrenament dels word embeddings van ser en primer lloc el CBOW (de l'anglès *continuous bag of words*) i després els salt-grames [12].

La diferència principal entre aquests dos models és que amb el CBOW la xarxa neuronal utilitzada infereix la representació d'una paraula prenent com a entrada el context en que es troba, mentre que amb els salt-grames la xarxa neuronal rep com a entrada la representació actual de la paraula i tracta de deduir el context en el que es troba a la capa de sortida. Aquesta diferència es pot veure gràficament a la figura 6.1.

Com es pot deduïr de la figura, el CBOW és un model molt similar als n-grames ja explicats, amb la significant diferència que les paraules no presenten un ordre específic, sinó que simplement es té en compte el conjunt. Els salt-grames també són similars als n-grames, però prenen el nom del fet que poden "saltar-se"algunes paraules. Per exemple, a la frase "*mai sé què dius*", utilitzant el model CBOW i conjunts de dues paraules, obtenim els grups "*mai sé*", "*sé què*" i "*què dius*". Utilitzant grups també de dues paraules, però ara amb salt-grames i saltant fins a dues paraules paraules, obtindríem "*mai sé*", "*mai què*", "*mai dius*", "*sé què*", "*sé dius*" i "*què dius*".

Tal com es pot veure d'una mateixa frase s'estan extraient més bi-grames, el que ha demostrat ser una bona solució al problema d'escassetat de dades en l'entrena-

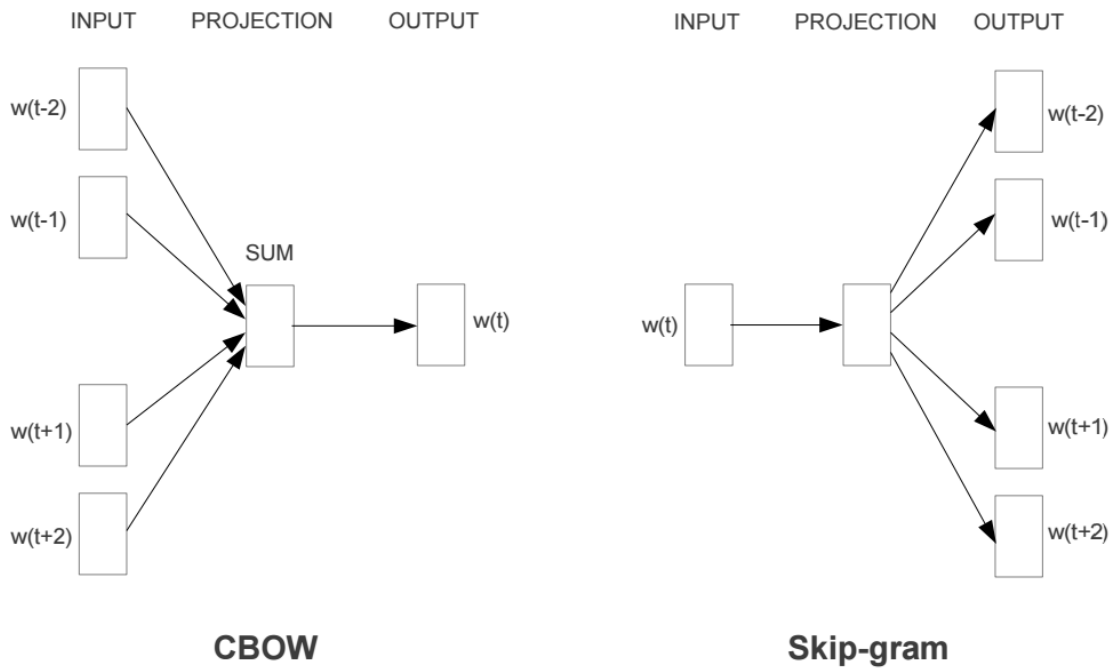

Figura 6.1: Representació de les arquitectures CBOW i salt-grames tal com es presenten a [14]. *w(t)* fa referencia a la paraula actual. Les altres paraules del context s'indiquen en base a *t*.

ment de models de llenguatge en el camp del processament de llenguatge natural [7].

Tot i això, cal fer notar que si bé als salt-grames les paraules poden tenir un ordre específic, tal com als n-grames, a l'hora d'aplicar-los als word embeddings aquest ordre es perd, ja que la representació de les paraules es fa en forma de vectors numèrics, no grups de paraules.

En conclusió, els word embeddings permeten l'extracció d'aquests vectors numèrics a partir de l'entrenament de xarxes neuronals, l'estructura de les quals varia en funció del model utilitzat per a l'entrenament. Aquests models, el CBOW i els salt-grames, també tenen alhora una sèrie de paràmetres que poden variar. Un dels conceptes més important és el de la "finestra"del model, que és el nombre de paraules (també anomenat çontext") que es tenen en compte en l'entrenament de les representacions.

El tipus de xarxes neuronals utilitzades en aquest procés han estat predominantment les xarxes neuronals recurrents, tot i que diversos grups han experimentat amb xarxes neuronals de diferents tipus, recentment fins i tot amb xarxes neuronals recurrents BLSTM (de l'anglès "*bidirectional long short-term memory*") [22]. Aquestes noves aproximacions estan donant resultats prometedors, i és d'esperar que es continuï investigant més en aquesta línia.

6.2 Representació vectorial de les paraules

En el cas dels n-grames, la informació es pot emmagatzemar com a seqüències de N paraules i una probabilitat associada. Els word embeddings, en canvi, associen un vector a cada paraula d'un idioma. En aquesta secció s'expliquen les característiques i interpretació d'aquests vectors, i es posen en context en comparació amb d'altres mètodes d'aprenentatge automàtic o models de reducció de dimensionalitat amb els que els word embeddings guarden certa relació.

Idea bàsica

Per a aquelles persones familiaritzades amb la interpretació dels vectors multidimensionals com a direccions, la representació dels word embeddings pot resultar prou entenedora. Tot i això, per a moltes persones aquesta representació de paraules com a vectors pot ser bastant més confusa.

Els vectors generats pels word embeddings poden tenir una alta dimensionalitat. Habitualment les representacions entrenades tenen entre cent i cinc-centes dimensions. Aquest nombre de dimensions es pot decidir en el moment de l'entrenament com si es tractés d'un paràmetre més, i és de fet el nombre de neurones amb el que es configurarà la capa oculta de la xarxa neuronal utilitzada per al procés d'entrenament del model.

Cadascuna de les dimensions es correspon a un valor numèric entre -1 i 1 del vector resultant per a una paraula. Tècnicament aquest rang de valors no és estrictament necessari, ja que el que interessa d'aquests vectors és la direcció, però és habitual treballar amb els vectors ja normalitzats. A la figura 6.2 podem veure un exemple de la forma que tenen els vectors generats.

```
etéreo:
-0.281204, 0.108265, 0.333351, -0.11214, 0.17311, -0.1578,
-0.045731, -0.03492, 0.030105, 0.051246, 0.19266, 0.29477,
-0.194046, 0.189661, 0.115785, -0.54918, -0.2216, ...
```

Figura 6.2: Exemple de l'inici d'un vector generat amb word2vec.

Els vectors generats individualment no contenen gaire informació; per treure'n profit els hem de comparar amb altres vectors, altres direccions, altres paraules del mateix espai. L'espai en que es troben aquests vectors es pot interpretar com una hiperesfera –de tantes dimensions com tingui l'espai–, amb els vector apuntant a diferents direccions. Aquestes direccions apunten a les representacions n-dimensionals dels contextos en els que apareixen les paraules. Expressat d'una altra manera: el vector d'una paraula representa una aproximació del seu context habitual (en el marc de l'espai n-dimensional de l'entrenament).

Així doncs, veiem que les representacions vectorials simplement són la projecció d'aquests contextos a un espai d'una dimensionalitat determinada.

Reducció de dimensionalitat

D'una banda hem vist que els word embeddings fan ús de les xarxes neuronals, però d'una forma molt interessant. No utilitzen una única xarxa neuronal com a classificador o similars, sinó que usen les xarxes neuronals per obtenir tantes representacions com paraules puguem tenir en un corpus de text. A més, tal com hem vist a l'última secció, aquesta representació és una projecció a un espai amb un nombre de dimensions habitualment molt elevat. Però de fet en aquest procés, tot i tenir un gran nombre de dimensions s'està fent una reducció de dimensionalitat.

Els mètodes de reducció de dimensionalitat són molt utilitzats en diverses disciplines, i el processament de llenguatge natural no n'és una excepció. Un dels mètodes més coneguts és la descomposició en valors singulars (abreviat SVD en anglès). En el camp del processament d'imatge, per exemple, és possible utilitzar la descomposició en valors singulars per comprimir imatges, tal com es mostra a la figura 6.3. La idea és eliminar els valors singulars menys rellevants projectant els N colors de la imatge a un nou espai amb M colors, sigui $M < N$.

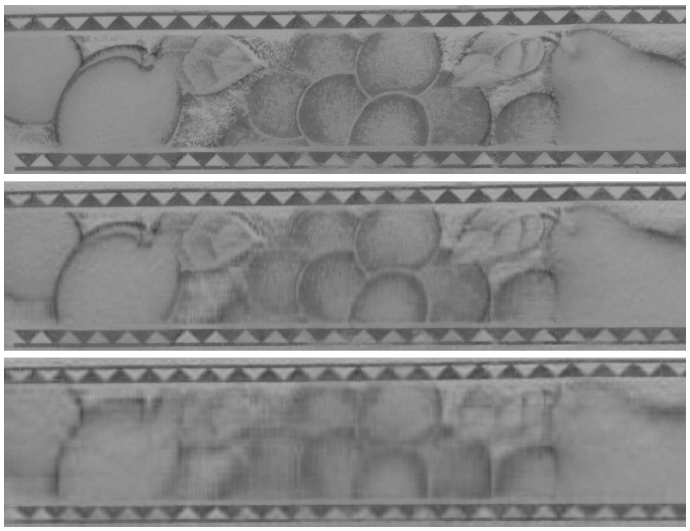

Figura 6.3: Mostra de compressió d'imatges usant SVD. A dalt es mostra la imatge original, la segona imatge s'ha reduït a 30 valors singulars, i la última a tan sols 15.

La imatge en realitat és només una matriu de píxels amb una sèrie de valors en una escala de grisos, i aquesta és la interpretació matemàtica de la SVD que ens interessa. En el cas dels word embeddings, també tenim una matriu formada per les representacions de totes les paraules del nostre model.

En el camp del processament de llenguatge natural, el mètode de descomposició de valors singulars s'utilitzava en tècniques com l'anàlisi semàntic latent, que de forma similar als word embeddings permet obtenir representacions vectorials de les paraules. Aquesta tècnica ja existia a finals dels anys 80, però no va ser fins més tard que es va popularitzar en el camp del PLN una versió millorada anomenada anàlisi semàntic latent probabilístic (PLSA en anglès) [8]. Així doncs, veiem com la idea dels word embeddings no és completament nova, sinó que combina mètodes perfeccionats en els últims anys (xarxes neuronals, representacions vectorials, reducció de dimensionalitat).

Similitud entre paraules

Un cop donades aquestes representacions en forma de vectors de les paraules, ens podem preguntar com les podem comparar.

Seguint la interpretació dels vectors com a direccions, comparar dues paraules és tan senzill com comparar les direccions dels vectors que les representen. Aquesta comparació es realitza tradicionalment computant la similitud del cosinus. Donats dos vectors A i B de la mateixa longitud, sigui aquesta N, la similitud del cosinus es defineix com:

$$similitud = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

On A_i i B_i són les components i-éssimes dels vectors. Aquesta expressió és deriva del producte escalar dels vectors dividit pel producte del mòdul dels vectors:

$$similitud = \frac{A \cdot B}{\|A\|\|B\|}$$

En altres paraules, aquesta similitud del cosinus és realment el cosinus de l'angle que formen aquests dos vectors. Si els vectors són perpendiculars, aleshores la seva similitud serà zero. Si els vectors apunten a la mateixa direcció o direccions molt semblants i tenen el mateix sentit, la seva similitud serà propera a u. Finalment, si apunten en sentits oposats, la seva similitud serà negativa.

Així, calcular les similituds entre paraules és molt senzill i computacionalment eficient, ja que tan sols es requereixen operacions amb vectors molt simples.

6.3 Entrenament

Per aconseguir les millors representacions possibles de les paraules, l'entrenament dels models generats amb els word embeddings és clau.

L'entrenament bàsicament pren com a entrada un corpus de text no anotat, elabora una llista del vocabulari observat en el corpus i després passa a entrenar la xarxa neuronal per a cada paraula, utilitzant els contextos en els que la paraula apareix. El procés de fet és més complex, i es poden realitzar diverses iteracions o esperar a que les representacions de les paraules es tornin estables. El mètode exacte utilitzat també varia en funció de si s'utilitza el model CBOW o els salt-grames.

En el cas del CBOW, el model és molt més ràpid d'entrenar, però també s'obtenen representacions de lleugerament menor qualitat que amb els salt-grames. Els salt-grames, tal com ja s'ha explicat en seccions anteriors, extreuen majors conjunts de paraules per a una frase que el CBOW: aquest és un dels motius pel qual l'entrenament és més lent, però les representacions apreses acostumen a ser també de major qualitat. Un altre dels motius és que la funció objectiu a maximitzar per als salt-grames considera de l'ordre de tants contextos com paraules hi ha del vocabulari. Això és computacionalment molt ineficient, i és per aquest motiu que cal utilitzar tècniques com el softmax jeràrquic (de l'anglès *hierarchical softmax*) o el mostreig o *sampling* negatiu [12]. Aquestes tècniques es basen en el mostreig aleatori de contextos: en lloc de considerar tots els contextos possibles, se n'utilitzen mostres aleatòries, reduint així considerablement el temps d'entrenament dels models i mantenint una bona qualitat de les representacions obtingudes. Tot i així, hi ha cert debat [6] sobre la correctesa matemàtica del mètode de mostreig negatiu, introduït per Mikolov i el seu grup de recerca a Google el 2013.

Un altre punt important és que els word embeddings realment no consideren paraules com a tals, sinó símbols. Cada símbol és una cadena de caràcters, i els delimitadors d'aquests símbols són els espais entre paraula i paraula. Això vol dir que es poden obtenir representacions vectorials fins i tot d'elements com emoticones o signes de puntuació. A més, aquest tractament de les paraules com a símbols implica que cal fer un preprocés del text abans de tractar d'entrenar els models. Aquest preprocés bàsicament consisteix a separar paraules, signes de puntuació, emoticones... de forma similar al que fan els tokenitzadors.

Altres aspectes a tractar durant el preprocés dels textos són:

- **Majúscules**: les paraules en majúscules i minúscules es consideren símbols diferents, i per tant cal anar amb compte al tractar amb elles. En la majoria dels casos, resulta convenient tractar totes les paraules en minúscules i avaluar més endavant si les paraules han d'aparèixer en majúscules o no, utilitzant mòduls de reconeixement de noms propis, etc.

- **Compostos**: quan trobem expressions com "*xarxes socials*", si volem man-

tenir l'expressió com a un únic compost, s'haurà de preprocessar el text i transformar l'expressió en "*xarxes_socials*" (convertint-la en un únic símbol).

- **Formes verbals**: les formes verbals en certs idiomes poden tenir un gran nombre de variacions, el que provoca que moltes d'elles apareguin de forma molt infreqüent o directament no apareguin en els corpus de text d'entrenament. Una possibilitat consisteix a realitzar un preprocés que redueixi les formes verbals a la seva arrel, però la conveniència de fer això depèn molt de la tasca per a la que es vulguin utilitzar els word embeddings i la mida del corpus d'entrenament.

Tal com es pot deduir de les explicacions anteriors, una limitació dels word embeddings és que no poden tractar eficientment la polisèmia de les paraules, ja que les paraules polisèmiques segueixen tenint la mateixa forma per a les seves distintes accepcions.

Finalment, un altre punt interessant és que el tipus de corpus donat per a l'entrenament pot tenir una influència significativa sobre les representacions finals obtingudes. Ja s'ha mencionat anteriorment que en certes ocasions pot resultat convenient utilitzar corpus de text de menor mida però que tractin temes o camps específics. Aquesta estratègia es pot portar encara més lluny, com va fer un grup de recerca de l'Institut Tecnològic Toyota, que va aconseguir entrenar els word embeddings per a utilitzar-los en la detecció d'antònims [16].

6.4 Propietats

Un cop estudiats els word embeddings podem enumerar les seves propietats més rellevants i veure alguns dels resultats obtinguts en tasques de similitud entre paraules.

Propietats

Com hem pogut veure, la representació en forma de vector de cada paraula indica el context en el que aquesta paraula tendeix a aparèixer.

Hi ha molts tipus de paraules que tenen contextos semblants a d'altres. D'una banda tenim els sinònims: és d'esperar que paraules que es puguin utilitzar de forma més o menys intercanviable apareguin en contextos similars. Això és totalment cert, però s'ha d'anar en compte a l'hora d'extrapolar, ja que els antònims també són paraules que apareixen freqüentment en contextos similars. Els hipònims i hiperònims són un altre exemple de grups de paraules que tindran representacions similars. Finalment, també es pot esperar que variacions de les paraules (per exemple

variacions en nombre o gènere) apareguin en contextos semblants. Podem veure exemples de tot d'això a la taula 6.1.

frecuente	
Paraules similars	**Similitud del cosinus**
infrecuente	0.758609
habitual	0.740156
usual	0.72706
raro	0.690275
común	0.629933
frecuentes	0.592098
prevalente	0.579058
recomendable	0.573705
esporádico	0.566519
probable	0.557844

Taula 6.1: Paraules similars a "*frecuente*", extretes d'un model entrenat amb la Wikipedia en castellà. La dimensionalitat dels vectors és 200, la mida de la finestra 5 i el model utilitzat els salt-grames.

Una particularitat interessant és que a diferència dels n-grames, els models generats amb word embeddings poden tenir mides de finestra molt elevades sense que augmenti l'espai requerit per emmagatzemar-los. Així, tot i que els word embeddings poden tenir el desavantatge de no mantenir informació sobre l'ordre de les paraules, poden ser un model molt més adequat per representar contextos amplis.

Una propietat negativa que comparteixen amb altres models de llenguatge, és que les paraules més freqüents d'un idioma no acostumen a tenir representacions gaire informatives, ja que els contextos en els que apareixen són molt variats, i per tant, també difusos. En canvi, paraules que s'usen únicament en contextos molt específics acostumen a donar informació molt més precisa.

Malgrat tot, els word embeddings poden obtenir representacions de molt bona qualitat de les paraules i arribar a capturar relacions prou subtils quan els termes són prou específics. A la taula 6.2 es mostra un altre exemple de similitud de paraules, en aquest cas amb xarxes socials. Aquest exemple és particularment interessant perquè es veu com al tractar amb termes més específics, les similituds també arriben a tenir valors més elevats. Un altre detall interessant és que, al tractar les paraules com a símbols, el model ha après una representació errònia. La xarxa social Twitter apareix dos cops, amb el nom escrit incorrectament en la seva segona aparició a la llista.

Finalment, l'última propietat dels word embeddings és que l'espai vectorial en el que es representen les paraules manté la linealitat, de forma aproximada. Això vol dir que es possible, entre d'altres, utilitzar la relació entre dues paraules per calcular una analogia amb una tercera. Per exemple, podem calcular analogies del tipus "A és a B el que C és a D" prenent els vectors de les paraules i operant amb

facebook	
Paraules similars	**Similitud del cosinus**
twitter	0.914864
myspace	0.863881
instagram	0.843787
tumblr	0.82005
youtube	0.80511
linkedin	0.790852
snapchat	0.770022
flickr	0.765552
twiter	0.764362
vimeo	0.742815

Taula 6.2: Paraules similars a "*facebook*". El model utilitzat és el mateix que el de la taula 6.1.

ells com: $vec(D) = vec(C) + (vec(B) - vec(A))$. Es poden veure alguns exemples típics d'analogies a la taula 6.3.

A és a B el que C és a ?			
A	**B**	**C**	**?**
madrid	españa	parís	*francia*
parís	francia	londres	*inglaterra*
londres	inglaterra	tokio	*japón*

Taula 6.3: Exemple amb diverses analogies. La columna *?* és la columna calculada a partir dels word embeddings. El model utilitzat és el mateix que el de la taula 6.1.

En aquest cas s'utilitzen noms de països i les seves capitals ja que són termes específics i les seves representacions són molt precises, però també es possible tractar de fer analogies canviant el gènere i/o nombre d'una paraula, entre d'altres.

El mètode no és inequívoc, però els resultats obtinguts acostumen a ser prou raonables. Per exemple, en intentar fer l'analogia "*vídeo* és a *youtube* el que *foto* és a ?", els primers resultats obtinguts són "*foto*" i "*youtube*" de nou, però a continuació apareixen immediatament "*facebook*", "*instagram*" i "*flickr*" (Instagram i Flickr són xarxes socials orientades específicament a compartir fotografies pròpies, i són probablement les respostes esperades per a l'analogia).

6.5 Word2Vec

Durant la memòria del projecte ja s'ha esmentat en diverses ocasions el treball de Mikolov i el seu grup de recerca a Google estudiant els word embeddings. El seu

treball inclou nombroses millores a l'eficiència computacional i la qualitat de les representacions obtingudes amb aquest model semàntic de llenguatge, però una de les seves aportacions més importants és una implementació en codi obert dels word embeddings, anomenada *word2vec*.

Word2vec és popularment reconeguda com una de les millors implementacions dels word embeddings disponibles a la xarxa, tot i que existeixen d'altres implementacions. Word2vec està implementat en C, però hi ha implementacions similars en Java[1] i Python[2].

En aquest projecte s'ha utilitzat word2vec per a l'entrenament dels models utilitzats, però s'ha implementat una interfície pròpia per poder treballar amb els models un cop aquests ja han estat generats. D'aquesta manera es poden utilitzar els models de forma còmoda i ben integrada a FreeLing. Els detalls sobre la implementació es donen a la secció corresponent del capítol següent.

[1]Implementació de word2vec en Java accessible a `https://deeplearning4j.org/word2vec`

[2]La llibreria Gensim [19] conté entre d'altres tècniques relacionades amb el processament de llenguatge natural una implementació dels word embeddings.

7. *Implementació del projecte*

En aquest capítol es descriuen els diferents mòduls i sistemes implementats en el projecte de forma detallada, explicant també alguns dels problemes trobats així com les decisions preses per resoldre'ls. Tot el codi del projecte pot trobar-se a `https://github.com/sergillamas/corrector`.

La major part del codi està implementada sobre FreeLing, tot i que hi ha alguns fitxers relacionats amb el processament de text per a l'entrenament dels word embeddings o l'ús de l'API de Twitter que en són independents.

Tot el codi de FreeLing està implementat en C++, utilitzant moltes funcionalitats de C++11. Fora de FreeLing, s'han escrit i utilitzat alguns altres programes en Python, especialment els relacionats amb l'extracció i processament de dades de Twitter i Wikipedia.

7.1 Word embeddings

L'ús dels word embeddings es divideix fonamentalment en dues parts: l'entrenament dels models i el seu ús per a calcular similituds, analogies, i d'altres operacions amb els vectors que representen les paraules.

La idea inicial era portar les dues parts del codi directament a FreeLing, en concret utilitzant el codi de la implementació de word2vec. En mirar el codi en detall, però, vaig veure que el sistema per carregar models i realitzar operacions amb ells a word2vec deixava molt que desitjar. Bàsicament tenia tres problemes principals:

- **Llegibilitat**: en el codi es declaraven una sèrie de variables amb noms genèrics (a, b, c, etc.) que es reutilitzaven contínuament per a diferents propòsits. El codi tampoc contenia cap comentari que indiqués el que estava passant.

- **Eficiència**: les estructures de dades utilitzades en el programa no tenien cap forma d'accés eficient a la representació d'una paraula, així que cada cop que

es feia una consulta, la paraula consultada es cercava linealment a l'array en que estaven emmagatzemades les paraules.

- **Funcionalitats**: l'única operació que permetia fer el programa era la de trobar similituds entre paraules. De fet aquesta és la operació més important a realitzar amb els word embeddings, però jo volia tenir més control i possibilitats a l'hora de treballar amb les representacions de les paraules.

Tal com es pot veure, el programa realment no estava orientat a ser una interfície per als word embeddings, sinó únicament un programa d'exemple, i per tant era necessari o com a mínim molt aconsellable reescriure una interfície des de zero, ja que la complexitat de fer-ho tampoc era gaire elevada.

L'altre punt era l'entrenament dels models. En aquest cas, word2vec conté un únic fitxer que permet entrenar models, que no utilitza cap llibreria ni té dependències inusuals, i que en executar-se per línia de comandes dóna una bona quantitat d'informació sobre el procés d'entrenament. Així, hauria estat possible moure aquest codi a FreeLing de forma senzilla, però vaig considerar que no tenia sentit fer-ho. En primer lloc, el procés d'entrenament només requeria aquest fitxer, i per tant no hi ha gaires formes més senzilles d'entrenar un model que fer servir aquest programa directament. En segon lloc, els únics canvis a realitzar en el codi en cas de integrar-lo a FreeLing consistien en fer una classe que encapsulés les funcions del programa original i altres modificacions menors per ajustar el codi a l'estil de FreeLing, però no hi havia cap tipus de modificacions relacionades amb la funcionalitat del programa. Finalment, ja existeixen un gran nombre de models entrenats que es poden descarregar directament des d'internet, i per tant aquest entrenament dels models ni tan sols és essencial a l'hora de fer proves o experiments senzills amb els word embeddings.

Interfície per als word embeddings

La interfície implementada per als word embeddings segueix l'estil d'altres mòduls de FreeLing. Consisteix en dos fitxers, la capçalera (`word_vectors.h`) i la implementació (word_vector.cc) de la classe implementada. Aquests fitxers estan ubicats també en carpetes diferents, òbviament mantenint l'estructura pròpia de FreeLing.

Inicialització de la classe i manipulació dels models

La inicialització de la classe només requereix la ruta al fitxer que conté el model generat pels word embeddings i un paràmetre addicional indicant l'idioma del model. Aquest últim paràmetre de fet es pot ometre, ja que no és necessari per a cap funció de la classe que operi amb les representacions de les paraules.

La idea d'afegir informació sobre l'idioma del model sembla bastant lògica a FreeLing, ja que la llibreria es capaç de treballar amb diversos idiomes. El motiu pel que aquesta informació no s'ha incorporat al propi fitxer del model entrenat és que originalment aquesta informació no apareixia als models. L'estructura original d'aquests fitxers s'ha convertit pràcticament en un estàndard a l'hora de treballar amb ells, i per tant avui en dia és preferible mantenir la compatibilitat amb els diferents formats que modificar-los si no hi ha bones raons per fer-ho.

El format dels models generats pels word embeddings és una part molt important de la implementació d'aquesta classe, ja que la primera operació que cal fer en inicialitzar-la és llegir el model a partir del fitxer donat.

Els models generats pels word embeddings poden ser de dos tipus:

- **Format pla**: la representació de les paraules s'emmagatzema tal i com s'ha mostrat en figures anteriors, especificant una paraula seguida d'una llista de N nombres entre -1 i 1, on N és la dimensionalitat dels vectors. La primera línia del fitxer sempre conté el nombre de paraules del vocabulari i la dimensionalitat de les representacions utilitzades. A partir d'aquí, cada línia del fitxer conté una paraula amb la seva representació.

- **Format binari**: el format binari segueix la mateixa estructura del format pla, però en aquest cas el vector de nombres entre -1 i 1 que representa cada paraula està escrit al fitxer en binari, i per tant resulta impossible de llegir per a les persones.

Tot i que el format pla dóna una imatge molt clara i llegible de la informació real que conté el model, els fitxers generats són molt més grans que en format binari i també molt més lents de carregar, fàcilment d'un ordre de magnitud més lents. En particular, el format binari és especialment ràpid de llegir ja que els vectors de N nombres que representen una paraula es poden llegir en una sola operació com:

```
fread(punter_destí, sizeof(float), dimensionalitat, punter_fitxer)
```

Mentre que amb el format pla cal fer N lectures, sigui N la dimensionalitat dels vectors; una per cada component del vector. A continuació es mostren els càlculs per a una possible estimació de la mida en bytes dels models, i a la taula 7.1 se'n fa una petita comparativa.

Capçalera

< 50 bytes (mida del vocabulari i dimensionalitat dels vectors)

Format binari

< 10 bytes per paraula (en mitjana, assumint 2 bytes per caràcter)
+ 8 bytes per float * dimensionalitat

Format pla

< 10 bytes per paraula (en mitjana, assumint 2 bytes per caràcter)
+ 18 bytes per nombre * dimensionalitat

Mida vocabulari	**Dimensionalitat**	**Format binari**	**Format Pla**
300 000	100	231.79 MB	517.89 MB
300 000	200	463.58 MB	1.01 GB
300 000	300	695.37 MB	1.52 GB

Taula 7.1: Aproximació a les mides dels fitxers dels models de word embeddings.

En conclusió, podem veure com el format pla acostuma a requerir unes 2.25 vegades més espai que el format binari. També he realitzat algunes proves convertint alguns dels models que he utilitzat als dos formats, i les mides reals dels fitxers concorden amb aquests càlculs.

Tota aquesta lectura dels models ha estat de fet una de les parts més complicades d'implementar, ja que la implementació original de word2vec utilitza dades de tipus `char` i `string`, mentre que a FreeLing s'usen `wstring`. Això va implicar canviar lleugerament el mètode de lectura del fitxer, però el format binari en particular em va donar bastants problemes, ja que tot i aconseguir llegir el fitxer, inicialment utilitzava un mètode que resultava gairebé tant lent com la lectura del format pla.

El tipus de format utilitzat, binari o pla, es detecta en funció de l'extensió del fitxer. Si el fitxer té l'extensió `.bin`, el model es considera un model binari. En qualsevol altre cas, el model es carrega com un model en format pla, tot i que per al format pla és habitual usar l'extensió `.words`. La classe també incorpora funcions per convertir els fitxers d'un format a l'altre.

Finalment, a diferència de la implementació de word2vec, el model s'emmagatzema usant un diccionari, de forma que l'accés a les paraules i les seves representacions no té cost lineal sinó logarítmic respecte la mida del vocabulari del model.

Operacions bàsiques

La classe implementa totes les operacions bàsiques que es poden esperar per als word embeddings, que són la comparació de paraules usant la similitud del cosinus, el càlcul d'analogies, i la cerca de les N paraules més similars a una altra paraula donada.

Aquestes funcions s'implementen tant per paraules com per vectors. Normalment els word embeddings s'usen per comparar paraules que ja tenim al vocabulari, però quan fem analogies, per exemple, s'està calculant un vector intermedi que no té cap paraula associada però és una representació vàlida. La classe incorpora també operacions aritmètiques bàsiques amb vectors, i la idea es la de poder treballar amb paraules o les seves representacions (els vectors) indistintament. També és possible preguntar per l'existència de les paraules en el model, accedir directament a les seves representacions vectorials, o obtenir la llista completa del vocabulari.

Una de les operacions més costoses és la de trobar paraules similars a una altra paraula. Aquesta cerca està implementada de forma que es requereix un recorregut lineal per totes les paraules del vocabulari. Aquesta no és una operació utilitzada durant el procés de correcció, així que realment no és una operació crítica, però tractant-se les representacions de vectors, seria possible fer particions de l'espai per accelerar la cerca. Una altra possibilitat seria implementar mètodes indeterministes de cerca. De fet aquest és un tema que m'ha tingut intrigat durant tot el projecte, tot i que no m'he dedicat a provar d'implementar solucions ja que realment no les requeria.

7.2 Extracció de dades per proves i entrenaments

En aquest projecte hi havia una part important d'extracció de dades. D'una banda, necessitava grans corpus de text per a l'entrenament dels models, i de l'altra també m'interessava extreure tweets de Twitter per provar el funcionament del corrector amb exemples reals.

Extracció de text de Wikipedia

L'entrenament dels models de word embeddings requereixen corpus de text importants. Per a aquest projecte el corpus de text usat s'ha obtingut de Wikipedia.

Wikimedia, una de les fundacions creada per Jimmy Wales en el marc del projecte de Wikipedia, ofereix regularment versions de les wikipedies per a descarregar, accessibles a `https://dumps.wikimedia.org/`. Existeixen diversos tipus de ver-

sions: completes, parcials, incloent contingut multimedia o únicament text. Les versions completes amb contingut multimedia inclòs d'algunes wikipedies pot arribar a estar per sobre dels 20 terabytes. Si prenem únicament text, la mida està al voltant de les desenes de gigabytes. La Wikipedia en anglès, per exemple, està per sobre dels 50GB, mentre que Wikipedia en castellà està poc per sobre dels 10GB.

Per a aquest projecte vaig descarregar diverses versions de la Wikipedia en castellà, la més recent amb data del 10/20/2016, amb una mida de 2.6GB en format comprimit i 11GB un cop descomprimit. Tot i això aquesta mida és enganyosa, ja que un cop processat tot el text, la mida final va tornar a quedar en 3GB.

El processat del text bàsicament consisteix en eliminar tot el llenguatge de marques present, que és una combinació de XML i Wikicode, el llenguatge de marques propi de Wikipedia. Per a realitzar aquest procés vaig provar diversos programes, i finalment vaig optar per utilitzar *WikiExtractor*[1], disponible sota una llicència GNU GPL.

El propi text de la Wikipedia també es troba sota llicències similars, i el seu ús per a la mineria de dades tal com es realitza en aquest projecte està permès. Les condicions legals i llicències es poden trobar a la pròpia pàgina de Wikimedia, particularment a `http://dumps.wikimedia.your.org/legal.html`.

Un cop eliminat el llenguatge de marques, *WikiExtractor* deixa el text dividit en subcarpetes i fitxers més petits. Concatenar aquest fitxers a Linux és tan simple com utilitzar una línia com:

```
find .  -name 'wiki_*' -exec cat  + > ALL.txt
```

Després d'això, encara cal acabar de preparar el text, separant els símbols que es volen considerar durant l'entrenament dels word embeddings i aplicant totes les transformacions desitjades, com convertir tot el text a minúscules i eliminar o modificar les paraules o caràcters que no es vulguin tenir en compte durant l'entrenament. Un exemple poden ser els símbols gràfics en Unicode. Per a realitzar aquest procés vaig escriure un programa breu en Python anomenat `format_text.py`, que deixa el text completament preparat per utilitzar-lo en l'entrenament dels word embeddings. Aquest entrenament, tal com ja s'ha indicat prèviament, s'ha realitzat utilitzant word2vec.

Extracció de tweets de Twitter

Per extreure tweets i informació de Twitter he escrit dos programes breus en Python, `random_tweets.py` i `get_tweets_info.py`.

El primer, `random_tweets.py`, utilitza l'API de Twitter per extreure tweets aleatoris de Twitter en castellà. Realment Twitter no ofereix cap funció per a obtenir

[1] Accessible a `https://github.com/attardi/wikiextractor`

una mostra aleatòria de tweets en un idioma en particular, així que l'estratègia utilitzada és obtenir tants tweets com sigui possible en castellà. Twitter ofereix una funció que et permet obtenir els tweets més recents que contenen les paraules que li especifiquis. La idea és utilitzar una llista de les paraules més habituals del castellà per tal d'obtenir tots els tweets possibles.

Per funcionar el programa requereix un compte a Twitter. Cal utilitzar el compte per accedir a la secció de desenvolupadors de Twitter i crear una aplicació des d'allà. Un cop creada, es poden obtenir els *tokens* i claus necessàries per establir la comunicació amb l'API de Twitter. La meitat d'aquestes claus són confidencials, i és per això que no s'inclouen al programa les que he utilitzat personalment.

Aquest programa escriu tots els tweets obtinguts per pantalla (normalment redireccionats a un fitxer), incorporant tota la informació que es tingui disponible sobre els tweets, com poden ser l'autor, l'identificador del tweet, l'hora de creació, la ubicació des d'on s'ha enviat el missatge en cas que es tingui la informació, etc. Aquesta informació s'utilitza únicament com a entrada per al segon programa, i el fitxer s'esborra posteriorment per tal de complir les normes d'ús de l'API de Twitter, que no permet l'emmagatzemament de tweets de forma indefinida.

El segon programa, `get_tweets_info.py`, obté el text dels tweets obtinguts amb el primer programa i genera un fitxer en un format llegible. Aquest tipus de fitxers són els que he usat per elaborar jocs de prova per al corrector del projecte. De nou, els fitxers s'eliminen quan ja no s'estan utilitzant, o s'anonimitzen en cas que els vulgui utilitzar durant més temps, com per exemple per fer demostracions o re-avaluacions de les millores del corrector.

7.3 Mòdul de correcció

En aquesta secció es discuteix la implementació del mòdul principal del projecte, el mòdul de correcció, i s'expliquen algunes de les decisions preses per a la seva implementació en relació als problemes trobats en les proves realitzades.

Cal tenir en compte que el funcionament de FreeLing es basa en l'ús d'una sèrie de mòduls que analitzen diferents característiques d'un text de forma incremental. Els mòduls, per tant, es cridaran en funció de l'objectiu de l'usuari. Com ja s'ha explicat anteriorment, la primera part del procés de correcció consisteix en tokenitzar la frase i en generar alternatives. Aquests processos ja estan implementats a FreeLing, i queden fora d'aquest mòdul de correcció. Això vol dir que abans de cridar al mòdul de correcció, cal passar el text d'entrada pel tokenitzador, l'analitzador i el generador d'alternatives de FreeLing. Amb el codi del projecte ja s'incorporen els programes de prova que realitzen totes aquestes tasques, seguint el flux de treball natural a FreeLing.

Aquest mòdul de correcció també actua de forma similar, afegint la informació

calculada a la frase. Un cop calculades les millors seleccions d'alternatives, aquestes queden anotades en les pròpies alternatives. Així, cada alternativa té un atribut que indica les millors correccions de la frase en les que participa, si es que participa en cap.

Finalment, el mòdul de correcció és capaç de treballar en qualsevol idioma suportat per FreeLing, sempre que es compti amb els fitxers de configuració necessaris per generar les alternatives i els models de word embeddings entrenats per a aquell idioma en particular.

Inicialització del mòdul

FreeLing utilitza un sistema propi de fitxers de configuració que contenen tota la informació rellevant a l'hora d'inicialitzar un mòdul. Per exemple, l'analitzador de FreeLing es pot configurar per fer de forma opcional detecció de dates, nombres, noms propis, etc.

El mòdul de correcció ha estat implementat seguint el mateix model, i els fitxers de configuració contenen la següent informació:

- La ruta al model de word embeddings a utilitzar. Aquest paràmetre és necessari ja que el corrector ha d'inicialitzar la interfície dels word embeddings per poder treballar amb ells.
- L'idioma del text a corregir.
- Configuració addicional sobre els algoritmes utilitzats pel corrector. A través d'aquests paràmetres es poden configurar els mètodes d'avaluació i cerca en l'espai d'alternatives, explicats a la següent secció.

El temps que triga el mòdul en inicialitzar-se depèn bàsicament del que trigui el mòdul dels word embeddings en carregar el model donat. A partir d'aquí, en qualsevol moment es pot invocar el mètode `normalize` del corrector per corregir una frase.

Mètodes de cerca

Un cop es crida el mètode `normalize` comença el procés de correcció de la frase o frases donades. El primer que fa el corrector en aquest cas es guardar una còpia del text a corregir en un format més adequat per a la correcció. FreeLing fa servir llistes de frases per emmagatzemar els textos. Cada frase alhora està formada per paraules, que contenen informació sobre els anàlisis realitzats sobre elles, entre les quals tenim les alternatives generades per a les paraules incorrectes. Aquesta representació com

a llistes de llistes no és la més adequada pel procés de correcció, i és per això que s'extreuen únicament les dades que interessen i es copien en un vector de vectors. Més específicament, es un vector de parells de paraules i vectors d'alternatives:

```
typedef vector<pair<wstring, vector<freeling::alternative>>> alt_t;
```

Un cop es té aquesta estructura, es comença a explorar l'espai de possibles solucions. La idea original era utilitzar l'algoritme de Viterbi, un algoritme de programació dinàmica capaç de trobar el camí òptim en una seqüència de possibles estats, tal com es mostra a la figura 7.1. L'algoritme avalua les possibilitats per a cada nou estat, i guarda el millor resultat i camí fins a aquell punt.

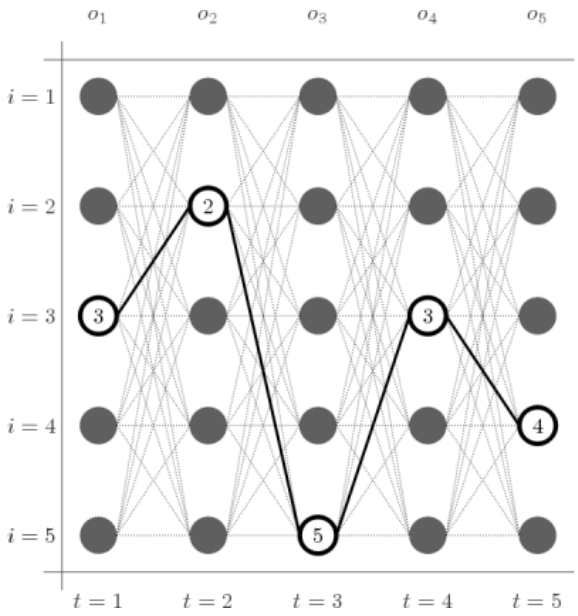

Figura 7.1: Representació gràfica d'una seqüència d'estats sobre la que s'ha aplicat l'algoritme de Viterbi. Imatge extreta de Wikipedia.

Tot i això, després de considerar diversos mètodes d'avaluació de les possibles solucions, va resultar que l'algoritme no es podia aplicar amb els word embeddings. Això passa perquè cada decisió presa sobre una alternativa afecta a la coherència o puntuació de la resta de la frase al complet.

En el cas dels n-grames, aquest problema no es dóna tant ja que normalment els n-grames utilitzats tenen contextos prou petits (considerem per exemple els 3-grames) i es poden prendre moltes de les decisions en cadena, tenint en compte només una petita part de la frase.

En veure això es van haver de considerar d'altres mètodes per a fer la cerca en l'espai de solucions.

Cerca exhaustiva

La primera solució òbvia és la cerca exhaustiva. Per cada paraula incorrecta, es limita el nombre d'alternatives considerades a 30. Algunes paraules tenen moltes menys alternatives, i d'altres en tenen moltes més. En particular, com més curtes són les paraules, més fàcil és generar-ne més alternatives. Donat aquest nombre de paraules, veiem com si tenim poques paraules a corregir realment es poden avaluar totes les combinacions de solucions possibles.

Per exemple, fins i tot si considerem que sempre tenim 30 alternatives per cada paraula, si només en tenim una a corregir, només hem d'avaluar 30 possibilitats. Amb dues paraules, el nombre de combinacions augmentaria a 900. Amb tres, a 27000. Experimentalment, he vist que fins a aproximadament 50000 combinacions el sistema és capaç d'avaluar-les ràpidament, normalment en menys d'un segon.

Tot i així la combinatòria explota molt ràpidament, i quan comença a haver-hi moltes paraules incorrectes el sistema es penja si s'utilitza la cerca exhaustiva. De totes maneres, aquest mètode em va permetre començar a provar el sistema prou aviat i realment segueix sent la millor opció quan el nombre de combinacions a explorar és prou reduït. De fet quan el nombre de combinacions és petit el mòdul ja s'encarrega d'utilitzar sempre aquest mètode automàticament.

Algoritme genètic

Per tractar de resoldre el problema que hi havia quan el nombre de paraules a corregir era elevat, vaig tractar d'implementar un algoritme genètic.

La idea d'utilitzar un algoritme genètic ve donada pel fet que per avaluar una potencial solució per a la correcció d'una frase necessitem tenir una solució completa. És a dir, hem d'haver decidit quina alternativa s'utilitzarà en cada posició. Per tant, d'entrada vaig apostar per utilitzar un algoritme genètic en lloc d'intentar utilitzar solucions constructives.

L'esquema bàsic del funcionament d'un algoritme genètic podria ser el següent:

1. Inicialitzar un conjunt de solucions.
2. Avaluar les solucions obtingudes.
3. Seleccionar i generar les solucions que passaran a la següent iteració de l'algoritme.
4. Aplicar d'altres operadors, com l'operador de mutació, per donar més aleatorietat i variació a l'algoritme.
5. Repetir el procés des del segon punt, a no ser que l'algoritme finalitzi.

En la terminologia tradicional per als algoritmes genètics, les solucions s'acostumen a anomenar ïndividusï les iteracions "generacions", seguint la idea del procés natural en el que es basa l'algoritme.

En la primera implementació que vaig realitzar, vaig tenir el problema que les solucions convergien massa ràpidament, després de molt poques generacions. Més tard vaig descobrir que el problema era el mètode de selecció utilitzat: estava ordenant les solucions per qualitat i eliminant la pitjor meitat. Aquest tipus de selecció era molt estricta i provocava que les millors solucions s'imposessin molt ràpidament sobre la resta, que no tenien temps de variar o canviar massa.

Per solucionar el problema vaig decantar-me per usar l'algoritme de selecció de la ruleta. Aquest algoritme permet seleccionar un individu de l'actual generació amb una probabilitat proporcional a l'aptitud de l'individu. El nom ve de les ruletes d'apostes. Aquestes tenen habitualment 37 nombres, del 0 al 36, i els nombres estan dividits en colors vermell i negre, a excepció del 0. En aquest cas, la probabilitat de que la bola caigui en una casella vermella és la mateixa que caigui en una casella negra. La probabilitat de que caigui a la casella del 0, en canvi, és molt menor. Per tant, després de provar sort 37 vegades els resultats que esperaríem serien d'aproximadament la meitat d'intents amb la bola caient a caselles vermelles, l'altra meitat a caselles negres, i potser alguna a la casella del 0. L'algoritme de la ruleta tracta de fer el mateix, seleccionant amb major probabilitat aquells individus més aptes.

La part important aquí és que el mètode és probabilístic i introdueix certa aleatorietat, el que permet que de tant en tant males solucions passin a la següent generació. Aquesta variabilitat és la que ajuda als algoritmes genètics a funcionar bé.

Per la part de generació de noves solucions, vaig utilitzar els operadors típics de recombinació. En alguns dels casos després de seleccionar un individu el feia passar directament a la següent generació, mentre que en d'altres casos seleccionava un segon individu i el recombinava amb el primer. La probabilitat de recombinació i selecció automàtica estan controlats per paràmetres, i les decisions es prenen de forma aleatòria en base a aquests paràmetres. Aquesta és una forma habitual d'implementar aquest pas.

Sobre la selecció de les solucions inicials, vaig començar amb solucions aleatòries, i després vaig aprofitar l'algoritme de la ruleta de nou per combinar les solucions aleatòries amb solucions en que es consideraven més aptes les alternatives amb menors distàncies d'edició a la paraula original.

Malgrat tots aquests canvis, si bé els resultats de l'algoritme van millorar, encara no vaig aconseguir obtenir bons resultats. La conclusió que n'he extret és que realment l'algoritme genètic amb els operadors de recombinació no és gaire adequat per la tasca. El motiu d'això és que l'algoritme parteix solucions per un punt determinar i les torna a combinar amb una altra solució. Aleshores, si tens dues bones solucions, combinar una part de solució bona amb una altra part de solució

bona pot donar una solució encara millor. De fet també es poden obtenir bones solucions combinant parts de solucions menys òptimes, però en general l'algoritme tendeix a anar fent aquest millora i recombinació progressiva de les solucions. Això amb les correccions no acaba de funcionar bé ja que encara que tinguis dues parts de frase que tinguin sentit, no sembla que la seva combinació sigui un bon heurístic per a generar frases completes amb sentit. A més, els màxims locals per al procés de correcció usualment són molt males solucions, ja que les frases gairebé sempre només tenen una única possible combinació de correccions que li donen sentit.

Tècniques d'avaluació de les solucions

Un cop donada una possible solució, és a dir, una sèrie de decisions sobre les alternatives escollides per a les paraules incorrectes del text, cal avaluar-la.

En aquest projecte s'han utilitzat els word embeddings per calcular la validesa de les solucions donades. Com ja s'ha explicat anteriorment, una de les avantatges dels word embeddings és que permeten representar contextos amplis d'una paraula de forma molt eficient. Models com els n-grames normalment es limiten a contextos molt reduïts, i només tenen en compte les paraules immediatament adjacents a la paraula a corregir.

Aquesta estratègia de fet dóna molt bons resultats, i s'ha utilitzat en castellà en un sistema que ha obtingut un 78.25% de precisió al Tweet-Norm [18]. Tot i això, una part important d'aquest projecte consistia en veure quins resultats s'aconseguien utilitzant word embeddings i considerant contextos més amplis.

La primera idea va ser simplement per provar els word embeddings i va consistir en calcular la similitud de les paraules de forma encadenada, comparant només una paraula amb la següent. Per aquest mètode de fet sí es podria haver usat l'algoritme de Viterbi, ja que realment no està considerant contextos amplis. En les proves inicials el mètode va ser un absolut fracàs, i els resultats obtinguts gairebé mai tenien sentit.

La següent estratègia va consistir en comparar totes les paraules de la frase entre elles. En aquest cas ja s'estava considerant un context de tota la frase, i els resultats van millorar lleugerament. Ara el sistema encertava de tant en tant. Com es pot observar, el context en aquest cas és de mida variable, mentre que el context amb el que s'entrenen els models no ho és realment. Vaig implementar una finestra de context per limitar l'avaluació, però els resultats no van millorar. En general les frases tractades de Twitter de totes maneres eren bastant curtes, i els contextos de les paraules a principi i final de frase també s'havien de retallar.

Una altra alternativa similar que vaig considerar va ser comparar la representació d'una paraula amb la suma dels vectors que representaven totes les altres. Aquest mètode de fet ha donat consistentment els mateixos resultats que l'anterior. En efecte, després de parar-hi més atenció, vaig veure que les ordenacions dels resultats

donades per $\sum_{\forall i \neq n} sim(V_n, V_i)$ i $sim(V_n, \sum_{\forall i \neq n} V_i)$ eren equivalents.

En aquest moment vaig veure que utilitzant els word embeddings en contextos amplis sense cap altre tipus d'informació els resultats no eren gens bons. Per aquest motiu, vaig prendre les distàncies d'edició i les vaig afegir a l'algoritme, ponderant els resultats dels word embeddings amb les distàncies d'edició de les paraules. Desafortunadament, aquest mètode tampoc va semblar millorar gaire els resultats obtinguts.

Un dels motius en part era que les distàncies d'edició estaven molt discretitzades de totes maneres, i per tant seguia tenint moltes alternatives a considerar que tenien el mateix valor de distància d'edició. A la secció següent s'expliquen algunes de les millores que vaig realitzar al sistema de càlcul de distàncies d'edició.

Finalment vaig mantenir el mètode de comparar totes les paraules entre elles, però vaig canviar la forma en la que usava els valors de les distàncies d'edició. En lloc de sumar els resultats dels word embeddings i les distàncies d'edició, vaig decantar-me per un altre sistema, en que la similitud de cada parell de paraules es multiplicava per la suma d'una probabilitat calculada en base a les distàncies d'edició de les paraules. Per exemple, les paraules que ja eren correctes a una frase tenien una distància d'edició 0, ja que no hi havia distància entre elles i les alternatives. En aquest cas, la seva probabilitat d'aparèixer a la frase en base a les distàncies d'edició és considera 1. En canvi, una alternativa a una paraula incorrecta té una probabilitat inversament proporcional a la seva distància d'edició. Les alternatives que tenen grans distàncies d'edició (estan més lluny de la paraula original) també tenen assignada una probabilitat més petita. Aleshores, quan comparo dues paraules utilitzant els word embeddings, la seva similitud es multiplica per la suma d'aquestes probabilitats.

Amb aquest sistema s'està donant més pes a les similituds que hi ha entre les paraules que semblen més probables –en base a les distàncies d'edició–, i en canvi es redueix el pes de la similitud entre paraules dubtoses.

Aquesta idea final és la que millors resultats ha donat d'entre els mètodes d'avaluació provats, tot i que els resultats segueixen quedant molt per sota dels obtinguts tradicionalment amb n-grames. Els resultats es poden veure al capítol 8.

Distàncies d'edició

FreeLing en el moment de calcular alternatives per a una paraula, ja calcula automàticament la seva distància respecte la paraula original. Tot i això, els resultats obtinguts estaven molt discretitzats i no es tenien en compte algunes operacions com la transposició de lletres.

La part positiva és que FreeLing ja utilitzava distàncies pròpies. El pes d'una

transformació com l'eliminació o substitució d'una lletra no tenia un valor de 1, sinó de 1000. Aquests valors a més es trobaven en fitxers de configuració fàcilment editables. Tot i això, els valors no estaven massa ben ajustats. Per exemple, FreeLing té un fitxer de configuració que considera errors típics de teclat, com els de substituir una lletra per alguna de les altres lletres adjacents en un teclat QWERTY. Però el pes d'aquest tipus de transformacions estava establert a 1000, quan hauria d'haver estat menor.

La primera part de la millora va consistir doncs en arreglar aquests valors. Els pesos han estat normalitzats considerant el pes d'una transformació com a 100. Els errors de teclat en canvi ara tenen pes 80. Lletres amb sons fonèticament semblants també es poden intercanviar amb pesos inferiors a 100.

La segona millora va ser afegir operacions que abans no existien. De fet la distància d'edició a FreeLing es calcula amb l'algoritme de Levenshtein tradicional. Per millorar lleugerament la situació vaig implementar la distància de Damerau-Levenshtein al propi mòdul del corrector, que considera les transposicions. Ara a l'inici del procés de normalització es comparen les distàncies d'edició i s'utilitza la distància de Damerau-Levenshtein en cas que aquesta resulti menor.

Avaluació dels resultats del corrector

El mòdul de correcció també té dues funcions destinades a comparar els resultats de les correccions amb els resultats correctes donats per l'usuari. Aquestes funcions ofereixen una forma automatitzada d'avaluar els resultats de la correcció quan es volen comparar diversos mètodes. Els resultats es mostren també per línia de comandes, en el format que es mostra a la figura 7.2.

```
### EVALUATION RESULTS ################################
# Number of sentences evaluated: 46

### Absolute results:
# Words corrected:          40/46
# Precision ceiling:        34/46
# Precision:                9/46

### Percentages:
# Words corrected:          86.9565%
# Words recalled:           73.913%
# Precision:                19.5652%
# Precision (over ceiling): 26.4706%
#######################################################
```

Figura 7.2: Resultats de la correcció realitzada sobre un joc de proves utilitzant únicament distàncies d'edició com a funció d'avaluació dels resultats.

Limitacions i altres problemes

Durant el procés d'implementació s'han trobat d'altres situacions problemàtiques o interessants que es recullen en aquest apartat.

Fins a aquest punt pot semblar que les alternatives a les paraules incorrectes sempre provinguin exclusivament de FreeLing, però existeixen dos casos en que no és així:

- Quan les paraules tenen una lletra repetida molts cops (per exemple: "*holaaaaaaa*"), el generador d'alternatives de FreeLing no és capaç de trobar la paraula original, ja que no contempla aquest tipus de casos. Pitjor encara, el generador pot penjar-se en el procés de buscar alternatives. Per exemple en aquest cas la lletra "a" per separat és una paraula correcta, i el generador comença a generar compostos del tipus "*hola_a_a_a_a_a*" i altres variacions fonètiques, com substituint aparicions de "*a*" per "*ah*". Per evitar aquests casos es fa un preprocés manual que en cas de trobar lletres repetides molts cops en una paraula, n'elimina totes les repeticions. En molts casos aquest procés és de fet el que genera l'alternativa vàlida (en l'exemple s'obtindria "*hola*").

- En alguns casos el diccionari de FreeLing no reconeix una paraula i la marca com a incorrecta, però el model de word embeddings sí conté aquella paraula en el seu vocabulari. Com hem vist en capítols anteriors, el vocabulari dels word embeddings no és molt fiable ja que pot aprendre paraules errònies. Tot i així, durant el procés de correcció si els word embeddings reconeixen la paraula incorrecta, aquesta s'afegeix a la llista d'alternatives com una possibilitat vàlida. En el cas de molts estrangerismes o noms propis escrits en minúscules, per exemple, aquesta estratègia ajuda a millorar els resultats de la correcció.

D'entre les limitacions del mòdul de generació d'alternatives, també hi ha el problema que no es contemplen les abreviatures. Aquestes són de fet bastant habituals en textos amb llenguatge xat, i és un punt que s'hauria d'implementar correctament si volguéssim millorar el corrector actual. En la implementació actual del mòdul de correcció es considera únicament una llista molt reduïda d'abreviatures habituals en castellà, que es mostren a la taula 7.2.

Una altra possible millora seria tenir en compte els compostos. FreeLing és capaç de generar alternatives en forma de compostos, però el procés de correcció actual no els té en compte, ja que en general entre aquests compostos hi ha molt soroll, i caldria implementar algun sistema per afinar més els pesos d'aquests compostos.

Finalment, el sistema també té problemes amb les majúscules. Això és així perquè no hi ha un mòdul que s'encarregui de passar correctament les paraules a majúscules/minúscules segons correspongui, i per tant es mira de mantenir les majúscules/minúscules tal com apareixen originalment a les frases.

Abreviatura no normativa	Correcció
q	que
xq/pq	porque
bss	besos
tb	también
xo	pero
x	por
bn	bien
xa	para
d	de
dnd	donde
sbs	sabes

Taula 7.2: Llista d'abreviatures no normatives no compostes considerades en el procés de correció.

7.4 Altres canvis a FreeLing

Tot i que en general tot el codi s'ha implementat en classes i fitxers nous, en alguns punts sí s'ha modificat el codi i els fitxers de FreeLing.

Un dels canvis ja explicats és l'ajustament dels pesos en els fitxers de configuració utilitzats per a calcular les distàncies d'edició. Un altre canvi necessari és el que s'ha fet en els Makefiles de FreeLing per poder compilar les noves classes implementades. Finalment, també s'ha afegit una nova classe anomenada `alternative`, que és utilitzada per guardar la informació sobre les millors alternatives seleccionades en el procés de correcció.

Anteriorment aquesta classe no existia ja que les alternatives eren un parell format per la pròpia alternativa i la seva distància d'edició respecte la paraula original. En aquest sentit, la nova classe permet un accés a aquests atributs i d'altres nous de forma més natural.

7.5 Programes de prova

Amb l'objectiu de poder provar tots els mòduls implementats s'han afegit també diversos programes de prova. De fet hi ha tota una carpeta amb programes que permeten utilitzar els mòduls implementats amb diferents objectius, i on s'inclouen també els fitxers de configuració i indicacions d'ús pertinents.

Els programes es descriuen molt breument a continuació:

- `corrector`: aquest és el programa que permet provar el mòdul de correcció.

L'usuari pot escriure frases per terminal i el programa retorna la versió corregida de la frase.

- `word_vec`: aquest programa serveix per provar la interfície dels word embeddings. Permet obtenir les paraules més similars a una paraula donada, comparar dues paraules en particular i calcular analogies.
- `evaluate`: de forma molt similar al programa del corrector, permet a l'usuari escriure frases, però requereix també les versions correctes. En escriure "EXIT", el programa finalitza i mostra els resultats de la precisió del corrector.
- `tweet_norm`: un programa d'avaluació similar a l'anterior, però que en aquest cas treballa amb el format utilitzat pel Tweet-Norm.
- `convert_model`: un petit programa que permet convertir models dels word embeddings d'un format a un altre. Si el format original és binari, genera un fitxer amb el mateix model en format pla, i al revés en el cas contrari.

8. *Anàlisi dels resultats*

En aquest capítol es mostren i s'expliquen els resultats obtinguts en les proves realitzades per avaluar el funcionament del corrector implementat.

8.1 Primer joc de proves

Per a les primeres proves vaig obtenir aproximadament una cinquantena de tweets en castellà i els vaig modificar perquè quedessin anonimitzats. Els tweets els vaig seleccionar perquè tinguessin una única paraula a corregir per frase, i que per tant a l'hora de corregir-los s'utilitzés el mètode de cerca exhaustiva en lloc de l'algoritme genètic. També vaig provar tots els mètodes d'avaluació que tenia. Els resultats obtinguts es mostren a la taula 8.1.

Mètode d'avaluació	**Precisió**	**Precisió (% rel.)**	**Precisió (%)**
Edit Distance	9/46	26.47%	19.56%
Similarity Next	1/46	2.94%	2.17%
Similarity All	10/46	29.4118%	21.7391%
Context Average	10/46	29.4118%	21.7391%
Probabilistic	18/46	52.9412%	39.1304%

Taula 8.1: Precisió en la tasca de correcció utilitzant diversos mètodes d'avaluació. A la columna "Precisió" es mostra el nombre de paraules corregides correctament, mentre que a "Precisió (% rel.)" es mostra la precisió respecte la màxima precisió assolible per a les alternatives donades.

La columna de la precisió relativa fa referència a la precisió del procés de selecció d'alternatives. En alguns casos la paraula correcta no es troba entre les alternatives donades, o la paraula ni tan sols ha estat marcada com a incorrecta. Aleshores no és possible seleccionar una alternativa correcta.

Com es pot veure en aquests resultats, la precisió de la correcció és baixa. El mètode d'avaluació "*edit distance*" fa referencia a l'avaluació de les potencials so-

lucions usant únicament les distàncies d'edició, i és el mètode que es pren com a referència.

En el cas del mètode "*similarity next*", que fa referència al mètode que calcula la similitud únicament entre paraules consecutives, els resultats no només són molt dolents, sinó que ni tan sols són millors que el que esperaríem d'un algoritme completament aleatori. De fet amb una mitja de 23 alternatives per paraula incorrecta, d'un algoritme aleatori esperaríem dues correccions encertades. El mòdul de correcció considera fins a 30 alternatives per paraula, però en bastants casos no n'arriba a tenir tantes.

La pregunta en aquest cas és per què els resultats són tan pobres. Pensar que una paraula tindrà certa similitud amb les paraules que la precedeixen i segueixen immediatament dins una frase d'entrada no sembla tan mala idea. Fins i tot pot semblar prou raonable pensar que poden tenir contextos més o menys similars. Un possible problema és que en un conjunt de tres paraules normalment una o dues aportaran poca informació, ja que es tractaran de preposicions, conjuncions, adverbis, etc. En aquest cas, els word embeddings difícilment obtindran informació prou indicativa del context tan reduït en que es troben.

A continuació tenim els mètodes de "*similarity all*" i "*context average*". Aquests són mètodes que tenen resultats equivalents ja que tenen en compte la similitud entre totes les paraules de la frase. Com es pot veure, els resultats d'aquests models estan lleugerament per sobre del mètode de la distància d'edició, però ni tan sols és de forma significant. Estadísticament, a més, donada la mida del fitxer de prova no podem dir que hi hagi cap diferència clara de resultats entre els tres mètodes.

El mètode probabilístic en canvi, anomenat així pel seu ús de les distàncies d'edició a l'hora de calcular una probabilitat inicial de que l'alternativa sigui correcta, sí dóna resultats clarament millors que les distàncies d'edició aïllades. Com ja s'ha explicat anteriorment, aquest mètode sí té certes avantatges al donar més pes a les paraules que ja sabem que estan amb tota seguretat a la frase, i donar-ne menys a aquelles paraules més dubtoses. D'aquesta manera les similituds donades pels word embeddings no s'apliquen de forma plana, sinó que s'ajusten millor a la informació que ja tenim de les frases.

8.2 Comparació amb Tweet-Norm

El joc de prova utilitzat per a la secció anterior ja dóna algunes idees sobre els resultats obtinguts amb la implementació actual del corrector, però el conjunt de tweets utilitzats és bastant limitat.

Tweet-Norm va oferir en el seu dia una col·lecció de 500 tweets per realitzar proves durant la tasca de correcció. Per motius de privacitat, tal com s'ha explicat en altres seccions, els tweets no es van distribuir directament, sinó que únicament es

distribueixen els seus identificadors. Quan un tweet és esborrat i es tracta de consultar a través del seu identificador, l'API de Twitter retorna un missatge indicant que el tweet ja no està disponible.

Dels 500 tweets originals, 116 havien estat eliminats quan vaig realitzar les proves, i per tant la col·lecció amb la que vaig avaluar el meu model contenia només 384 tweets. Tot i així, aquesta mostra de tweets és significativament més gran que la que havia recopilat jo mateix, i permet comparar els resultats obtinguts inicialment amb una altra mostra.

Per a aquesta prova vaig fer servir el model que donava millors resultats, el probabilístic, i en aquest cas sí s'utilitza també l'algoritme genètic com a mètode de cerca en els casos en que cal fer moltes correccions en un mateix tweet.

Un altre punt a considerar és que en els resultats d'aquestes proves s'ignora si les paraules estan en majúscules o minúscules, ja que com s'ha comentat anteriorment el mòdul de correcció no està preparat per realitzar aquesta tasca. En el Tweet-Norm originalment les majúscules i minúscules sí es van tenir en compte. Els resultats es mostren a la taula 8.2.

Mètode probabilístic	
Errors	68
Corr. Positives	148
Corr. Negatives	172
Precisió	38.14%

Taula 8.2: En aquesta taula es mostren el nombre d'errors comesos en el procés de correcció, el nombre de correccions encertades (positives) i el nombre de correccions equivocades (negatives).

El model de word embeddings usat per a realitzar aquesta avaluació va ser un model entrenat amb salt-grames, vectors de dimensionalitat 200, finestra 5 i el text de la Wikipedia sencera en castellà.

Com es pot veure en els resultats, hi ha un nombre important d'errors, el que indica que la detecció de paraules correctes/incorrectes encara es podria millorar.

Sobre la correcció de paraules pròpiament, el nombre d'errors segueix superant el nombre d'encerts. La precisió global del corrector es manté amb el que s'havia vist en els jocs de prova anteriors, quedant només un punt per sota. Tot i això, els resultats globals encara queden molt lluny dels obtinguts per altres grups al Tweet-Norm, com es pot veure a la taula 8.3.

Cal dir que hi ha diverses parts del corrector que realment no estaven totalment preparades per una tasca com la de Tweet-Norm, i això té un impacte important

Participant	Precisió
RAE	78.1%
Citius-Imaxin	66.3%
UPC	65.3%

Taula 8.3: Millors resultats al Tweet-Norm 2013.

sobre els resultats globals. La normalització de minúscules i majúscules, el reconeixement de noms propis i noms de companyies, les abreviatures o els compostos en són exemples. Però fins i tot obviant aquests factors, els resultats de la correcció usant word embeddings són modestos.

Hi ha diversos punts que expliquen aquests resultats. En primer lloc, la utilització dels word embeddings per capturar el significat de les frases en un context ampli pot ser adequada, però la seva aplicació per a la correcció de tweets no és tan oportuna. Els tweets són missatges curts que habitualment ja no tenen gaire context. En el cas del repte de completat de frases proposat per Microsoft [5], Mikolov va aconseguir millorar els resultats fins a un 58.9% de precisió. Però en aquest cas les frases acostumaven a ser molt més completes i riques, tal com es mostra a la figura 8.1.

1. I have seen it on him , and could _____ to it.
2. They seize him and use violence towards him in order to make him sign some papers to make over the girl's _____ of which he may be trustee to them.
3. My morning's work has not been _____ , since it has proved that he has the very strongest motives for standing in the way of anything of the sort.
4. It was furred outside by a thick layer of dust, and damp and worms had eaten through the wood, so that a crop of livid fungi was _____ on the inside of it.

Figura 8.1: Les quatre primeres frases d'un dels conjunts de preguntes del *Microsoft Research Sentence Completion Challenge*. Les frases incloïen cinc possibles respostes cadascuna.

Un altre dels problemes és la dependència dels word embeddings en altres mètodes. Per a aquest projecte m'ha resultat impossible elaborar un mètode de correcció que avalués les alternatives seleccionades sense usar també altres mètriques. La qualitat de les representacions apreses pels word embeddings és molt bona, però a l'hora de treballar amb contextos amplis, he necessitat utilitzar altres mètriques com a suport.

El fet que els word embeddings no considerin l'ordre de les paraules d'un context

també té significança. Aquesta falta d'ordre ja és apropiada a l'hora de treballar amb contextos amplis, però la informació de l'ordre en que es troben les paraules també sembla resultar molt útil en els n-grames. En el futur probablement seria interessant estudiar l'ús combinat dels diversos models per aprofitar les millors característiques de cadascun.

9. *Conclusions*

La correcció automàtica és una tasca extremadament complexa. Si bé les parts bàsiques d'un corrector es poden implementar de forma simple i existeixen molts errors que es poden corregir amb mètodes molt senzills, la dificultat de la correcció automàtica com a problema al complet és molt elevada. En l'actualitat els models de llenguatge com els n-grames ofereixen bons resultats per a la selecció de les millors alternatives d'una paraula incorrecta, però aquests models tenen limitacions importants al no tenir en consideració gran part de la informació donada en les frases.

Estadísticament els mètodes utilitzats avui en dia funcionen molt bé, i els correctors són capaços de corregir per sobre del 80% dels errors ortogràfics en diverses llengües [9][18]. Tot i això, els errors varien molt en complexitat, i cada cop resulta més difícil millorar els resultats sense utilitzar tècniques que considerin més informació sobre els textos.

És en aquest marc que aquest projecte valora l'ús dels word embeddings aplicats a la correcció automàtica, tractant de considerar contextos més amplis en el procés. D'aquesta manera, s'intenta aprofitar la informació semàntica de les frases al complet per orientar millor la selecció d'alternatives.

S'ha demostrat que aquesta aproximació al problema té sentit, tot i que els resultats obtinguts queden molt lluny dels de l'estat de l'art. Entre d'altres, s'observen els següents problemes:

- Els textos que requereixen més correccions acostumen a ser textos informals, en molts casos intercanvis de missatges curts o frases sense massa context. En aquestes situacions, els contextos amplis que poden representar els word embeddings no arriben a resultar tan útils.

- No s'han pogut trobar estratègies per avaluar les solucions usant únicament les representacions dels word embeddings, i ha resultat necessari utilitzar d'altres mètriques per a complementar el procés. Aquest fet no és intrínsecament negatiu, però vol dir que cal elaborar tècniques més complexes per a obtenir bons resultats.

- Els mètodes de cerca i optimització de solucions a usar amb aquest tipus

d'aproximacions són més complexos que quan s'usen models com els 3-grames. Els mètodes tradicionals com els algoritmes genètics provats tampoc semblen respondre de forma adequada.

- Per a aquest projecte en particular, el mòdul de correcció no era prou complet. La normalització de majúscules i minúscules, el reconeixement de noms propis, la identificació d'abreviatures i la correcció de paraules usant compostos són parts del procés que no han estat implementades però que resulten necessàries a l'hora de crear un corrector realment robust. A més a més, altres parts sí implementades com les distàncies d'edició i els mètodes de cerca encara es podrien optimitzar més. Un altre exemple seria el nombre d'alternatives considerades per cada paraula, que arriba fins a 30. Sota aquest nombre d'alternatives, els resultats dels word embeddings encara semblen prou bons.

Però no tot són punts negatius, i el projecte ha demostrat que realment la idea de combinar diversos models i mètriques per a l'avaluació de les possibles alternatives pot donar bons resultats. En particular, hi ha molts problemes amb els word embeddings que es podrien compensar utilitzant altres models de llenguatge o mètriques. Un exemple és la falta d'ordre en les paraules dels contextos per als word embeddings; en aquest cas, crec que combinar els word embeddings amb els n-grames podria donar bons resultats. Un altre exemple són les males representacions que obtenen les paraules més comunes. En aquest cas, tant els n-grames com la freqüència individual de les paraules es podrien utilitzar com a mètriques per tractar de compensar les representacions vagues apreses pels word embeddings. De fet, en general crec que els n-grames i salt-grames són els mètodes que haurien de continuar usant-se com a base per a la correcció de paraules, però que d'altres mètriques i models de llenguatge definitivament poden aportar valor si s'incorporen de forma adequada.

En conclusió, considero que queda molta investigació a realitzar en el camp del processament de llenguatge natural en global, a fi de desenvolupar tècniques que cada cop incorporin més informació sobre el llenguatge i que a més ho facin d'una forma més coherent.

Bibliografia

[1] Inaki Alegria et al. "Introducción a la Tarea Compartida Tweet-Norm 2013: Normalización Léxica de Tuits en Español". A: Tweet Normalization Workshop at SEPLN (Tweet-Norm). 2013.

[2] Yoshua Bengio et al. "A neural probabilistic language model". A: *journal of machine learning research* 3.Feb (2003), pàg. 1137 - 1155.

[3] Fred J. Damerau. "A Technique for Computer Detection and Correction of Spelling Errors". A: *Commun. ACM* 7.3 (mar. de 1964), pàg. 171 - 176. ISSN: 0001-0782. DOI: 10.1145/363958.363994. URL: http://doi.acm.org/10.1145/363958.363994.

[4] Stefan Evert. "Google Web 1T 5-grams Made Easy (but Not for the Computer)". A: *Proceedings of the NAACL HLT 2010 Sixth Web As Corpus Workshop*. WAC-6 '10. Los Angeles, California: Association for Computational Linguistics, 2010, pàg. 32 - 40. URL: http://dl.acm.org/citation.cfm?id=1868765.1868770.

[5] Chris J.C. Burges Geoffrey Zweig Christopher J.C. Burges. *The Microsoft Research Sentence Completion Challenge*. Inf. tèc. Des. de 2011. URL: https://www.microsoft.com/en-us/research/publication/the-microsoft-research-sentence-completion-challenge/.

[6] Yoav Goldberg i Omer Levy. "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method". A: *CoRR* abs/1402.3722 (2014). URL: http://arxiv.org/abs/1402.3722.

[7] David Guthrie et al. "A closer look at skip-gram modelling". A: 2006.

[8] Thomas Hofmann. "Probabilistic Latent Semantic Analysis". A: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. UAI'99. Stockholm, Sweden: Morgan Kaufmann Publishers Inc., 1999, pàg. 289 - 296. ISBN: 1-55860-614-9. URL: http://dl.acm.org/citation.cfm?id=2073796.2073829.

[9] Ning Jin. "NCSU-SAS-Ning: Candidate Generation and Feature Engineering for Supervised Lexical Normalization". A: *ACL-IJCNLP 2015* (2015), pàg. 87.

[10] Mikael Kågebäck et al. "Extractive summarization using continuous vector space models". A: *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*. Citeseer. 2014, pàg. 31 - 39.

[11] Jean-Baptiste Michel et al. "Quantitative analysis of culture using millions of digitized books". A: *science* 331.6014 (2011), pàg. 176 - 182.

[12] T Mikolov i J Dean. "Distributed representations of words and phrases and their compositionality". A: *Advances in neural information processing systems* (2013).

[13] Tomáš Mikolov. "Statistical Language Models Based on Neural Networks". Tesi doct. Ph. D. thesis, Brno University of Technology, 2012.

[14] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". A: *arXiv preprint arXiv:1301.3781* (2013).

[15] Robert C Moore i Chris Quirk. "Improved smoothing for N-gram language models based on ordinary counts". A: *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Association for Computational Linguistics. 2009, pàg. 349 - 352.

[16] Masataka Ono, Makoto Miwa i Yutaka Sasaki. "Word Embedding-based Antonym Detection using Thesauri and Distributional Information". A: *HLT-NAACL*. 2015.

[17] Lluís Padró i Evgeny Stanilovsky. "FreeLing 3.0: Towards Wider Multilinguality". A: *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*. ELRA. Istanbul, Turkey, mai. de 2012.

[18] Jordi Porta i José-Luis Sancho. "Word Normalization in Twitter Using Finite-state Transducers". A: *Tweet Normalization Workshop at SEPLN (TweetNorm)* (2013).

[19] Radim Rehurek i Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". A: *In Proceedings Or The LREC 2010 Workshop On New Challenges For NLP Frameworks*. 2010, pàg. 45 - 50.

[20] Xin Rong. "word2vec parameter learning explained". A: *arXiv preprint arXiv:1411.2738* (2014).

[21] Fermín Sánchez et al. *Guia y evaluación de la sostenibilidad en los Trabajos de Fin de Grado*. ISBN: 978-99920-70-10-9. DL: AND.92-2015. Andorra la Vella, Andorra: XXI Jornadas de Enseñanza Universitaria de la Informática, JENUI 2015, jul. de 2015, pàg. 34 - 41.

[22] Peilu Wang et al. "Learning distributed word representations for bidirectional lstm recurrent neural network". A: 2016.